

Distributed Supervisory Control of Discrete-Event Systems with Communication Delay

Ren yuan Zhang · Kai Cai · Yongmei Gan · Zhaoan Wang · W.M. Wonham

the date of receipt and acceptance should be inserted later

Abstract This paper identifies a property of delay-robustness in distributed supervisory control of discrete-event systems (DES) with communication delays. In previous work a distributed supervisory control problem has been investigated on the assumption that inter-agent communications take place with negligible delay. From an applications viewpoint it is desirable to relax this constraint and identify communicating distributed controllers which are delay-robust, namely logically equivalent to their delay-free counterparts. For this we introduce inter-agent channels modeled as 2-state automata, compute the overall system behavior, and then present an effective computational test for delay-robustness. From the test it typically results that the given delay-free distributed control is delay-robust with respect to certain communicated events, but not for all, thus distinguishing events which are not delay-critical from those that are. Another test is proposed to identify whether or not an uncontrollable event, which cannot be re-transmitted until its communication channel becomes available, will be blocked. The approach is illustrated by a workcell model with three communicating agents.

Keywords Discrete-Event Systems · Distributed Supervisory Control · Communication Delay · Delay-robustness

R. Zhang,
Xi'an Jiaotong University
E-mail: r.yuan.zhang@gmail.com

K. Cai
University of toronto
E-mail: kai.cai@scg.utoronto.ca

Y. Gan
Xi'an Jiaotong University
E-mail: ymgan@mail.xjtu.edu.cn

W.M. Wonham
University of Toronto
E-mail: wonham@control.utoronto.ca

1 Introduction

Distributed control is pervasive in engineering practice, either by geographical necessity or to circumvent the complexity of centralized (also called ‘monolithic’) control. Existing work on distributed supervisory control of discrete-event systems (DES) has focused on synthesis of local controllers for individual agents (plant components) such that the resulting controlled behavior is identical with that achieved by global supervision [1–6]. In these contributions, it is assumed that agents make independent observations and decisions, with instantaneous inter-agent communication. While simplifying the design of distributed control, this assumption may be unrealistic in practice, where controllers are linked by a physical network subject to delays. Hence, to model and appraise these delays is essential for the correct implementation of control strategies.

The communication problem in decentralized control of multi-agent DES has been discussed by several researchers. Taking delays into consideration, Yeddes et al. [7] propose a 3-state data transmission model, representing delays by timed events with lower and finite upper time bounds; these events are incorporated into the plant and specification automata, and the time bounds further restricted by a supervisor synthesis procedure; maximal permissiveness and nonblocking, however, are not guaranteed. In [8] Barrett and Lafortune propose an information structure model for analysis and synthesis of decentralized supervisory control, applicable in principle to the case of communication delays, but they assume that such delays are absent. For a limited class of specifications, Tripakis [9] formulates certain problems in decentralized control with bounded or unbounded communication delay, modeling the system with communication by automata with state output map. In this model the existence of controllers in case of unbounded delay is undecidable; however, it will be shown in this paper, that given controllers, designed under the assumption that communication delay is negligible, can solve the distributed control problem with unbounded communication delay under suitable conditions. Park and Cho [10] present a necessary and sufficient condition, in which ‘delay-coobservability’ is the critical concept, for the existence of a nonblocking decentralized supervisor that can achieve a given language specification under communication delays when the decentralized supervisor is assumed to have a conjunctive and permissive decision structure; however, the class of delay-coobservable languages is not closed under union; hence, maximal permissiveness is not guaranteed. Hiraishi [11] proposes an automaton formalism for communication with delay in decentralized control, and concludes semi-decidability of the controller design problem in the case of k -bounded delay and in case an observability condition holds for state-transition cycles.

In contrast with the foregoing contributions, distributed control problems with separately modeled communication channels having unknown unbounded delay, imposed on an existing distributed architecture known to be optimal and nonblocking for zero delay, seem not yet to have been investigated. Further discussion on communication channels will be presented in Sect. 3. In this paper we start from the DES distributed control scheme called ‘supervisor localization’ reported in [5, 6], which describes a systematic top-down approach to design distributed controllers which achieve optimal and global nonblocking supervision. Briefly, the initial control problem is the standard ‘Ramadge-Wonham’ (RW) problem [12–14]. Here the plant (DES to be controlled) is modeled as the synchronous product of

several DES agents (plant components), say **AGENT**₁, **AGENT**₂, ..., that are independent, in the sense that their alphabets $\Sigma_1, \Sigma_2, \dots$, are pairwise disjoint. In a logical sense these agents are linked by specifications **SPEC**₁, **SPEC**₂, ..., each of which (typically) restricts the behavior of an appropriate subset of the **AGENT**_i and is therefore modeled over the union of the corresponding subfamily of the Σ_i . For each **SPEC**_j, a ‘decentralized’ supervisory controller **SUP**_j is computed in the same way as for a ‘monolithic’ supervisor [12]; it guarantees optimal (i.e. maximally permissive) and nonblocking behavior of the relevant subfamily (the ‘control scope’ of **SPEC**_j) of the **AGENT**_i. In general it will turn out that the synchronous product of all the **SUP**_j is blocking (e.g. may cause deadlock in the overall controlled behavior); in that case one or more additional ‘coordinators’ must be adjoined to suitably restrict the decentralized controlled behavior (see [6] for an example). Techniques for coordinator design are available in the literature (e.g. [15–18]). On achieving satisfactory decentralized control we finally ‘localize’ each decentralized supervisor, including the coordinator(s), if any, to the agents that fall within its control scope; the algorithm that achieves this is detailed in [5], and we shall refer to it as *Localize*. The result of *Localize* is that each **AGENT**_i is equipped with local controllers, one for each of the **SPEC**_j whose scope it falls within; in that sense **AGENT**_i is now ‘intelligent’ and semi-autonomous, with controlled behavior **SUPLOC**_i, say, while the synchronous product behavior of all the **SUPLOC**_i is provably that of the monolithic supervisor for the RW problem we began with. Autonomy of the **SUPLOC**_i is qualified, in that normally the transition structure of each **SUPLOC**_i will include events from various other **AGENT**_k with $k \neq i$. The implementation of our distributed control therefore requires instantaneous communication by **AGENT**_k of ‘communication’ events (when they occur, in its private alphabet Σ_k) to **SUPLOC**_i so the latter can properly update its state. Think of a group of motorists maneuvering through a congested intersection without benefit of external traffic control, each instead depending solely on signals from (mostly) neighboring vehicles and on commonly accepted protocols. In our DES model each **SUPLOC**_i can disable only its private controllable events, in Σ_i , but the logic of disablement may well depend on observation of critical events from certain other **AGENT**_k, as remarked above. It is clear that if these communications are subject to indefinite time delay, then control may become disrupted and the collective behavior logically unacceptable. Our first aim is to devise a test to distinguish the latter case from the ‘benign’ situation where delay is tolerable, in the sense that ‘logical’ behavior is unaffected, even though in some practical sense behavior might be degraded, for instance severely slowed down¹.

As will be seen in Sect. 3, where the model of our communication channel is introduced, there is an implicit constraint that a channeled event (i.e. a communication event transmitted by a channel with indefinite delay) can occur and be transmitted only when its channel is available. As a consequence, an uncontrollable channeled event may or may not be blocked by its channel, the former case being undesirable. Our second aim is to distinguish these two cases; when an uncontrollable event is indeed blocked, we discuss how long it can be delayed.

¹ Similar issues are addressed in the literature on ‘delay-insensitive’ asynchronous networks; for the definition see [19] and for a useful summary [20].

We proceed to a formal review of distributed control by supervisor localization on the assumption of instantaneous inter-agent communication. Then we introduce inter-agent communication with delay, modeled by a separate logical channel for each delayed communication event (i.e. channeled event). As our main result, both a definition and a computational test are provided for ‘delay-robustness’ of the channeled distributed system with respect to an arbitrary subset of communication events. In addition, a verification procedure is proposed to identify whether or not an uncontrollable channeled event is blocked by its channel. These issues are illustrated by a workcell model with three communicating agents. Finally we present conclusions and suggestions for future work.

2 Preliminaries

2.1 Notation

Following the usage of [14] we recall various standard concepts and notation. Consider a system \mathbf{G} of n component DES $\mathbf{G}_i = (Q_i, \Sigma_i, \eta_i, q_{i0}, Q_{im})$, $i \in N := \{1, 2, \dots, n\}$, where Q_i is the (finite) state set, Σ_i is the (finite) set of event labels, $\eta_i : Q_i \times \Sigma_i \rightarrow Q_i$ is the state transition (partial) function, q_{i0} is the initial state, and $Q_{im} \subseteq Q$ is the set of marker states. Each event set Σ_i is partitioned as the disjoint union $\Sigma_i = \Sigma_{ic} \cup \Sigma_{iu}$ where Σ_{ic} (resp. Σ_{iu}) is the subset of controllable (resp. uncontrollable) events for \mathbf{G}_i ; the full event set for \mathbf{G} is the union $\Sigma = \cup\{\Sigma_i | i \in N\}$.

Let Σ_i^* denote the set of all finite strings of elements in Σ_i , including the empty string ε , and as usual extend the transition function η_i to $Q_i \times \Sigma_i^*$, by defining $\eta_i(q_i, \varepsilon) = q_i$, $\eta_i(q_i, s\sigma) = \eta_i(\eta_i(q_i, s), \sigma)$ for all $q_i \in Q_i$, $s \in \Sigma_i^*$ and $\sigma \in \Sigma_i$. The *prefix closure* of a language $L \subseteq \Sigma^*$ is defined as $\bar{L} = \{s \in \Sigma^* | su \in L \text{ for some } u \in \Sigma^*\}$. The *closed behavior* and the *marked behavior* of \mathbf{G}_i are defined respectively by $L(\mathbf{G}_i) = \{s \in \Sigma_i^* | \eta_i(q_{i0}, s) \text{ is defined}\}$ and $L_m(\mathbf{G}_i) = \{s \in L(\mathbf{G}_i) | \eta_i(q_{i0}, s) \in Q_{im}\}$.

As in [5, 6] we assume that the \mathbf{G}_i are *a priori* independent, in the sense that their alphabets Σ_i are pairwise disjoint. The system \mathbf{G} representing their combined behavior is defined to be their synchronous product [14] $\mathbf{G} = (Q, \Sigma, \eta, q_0, Q_m) = \text{Sync}(\mathbf{G}_1, \dots, \mathbf{G}_n)^2$. The closed behavior and marked behavior of \mathbf{G} are $L(\mathbf{G}) = \|\{L(\mathbf{G}_i) | i \in N\}$ and $L_m(\mathbf{G}) = \|\{L_m(\mathbf{G}_i) | i \in N\}$ where $\|\$ denotes synchronous product of languages [14]. Assume each \mathbf{G}_i is trim (i.e. reachable and coreachable, see [14]); then by independence, \mathbf{G} is trim, i.e., $\bar{L}_m(\mathbf{G}) = L(\mathbf{G})$ [14].

Let $\Sigma_o \subseteq \Sigma$ be a subset of events thought of as ‘observable’. We refer the reader to [14] for the formal definition of natural projection $P : \Sigma^* \rightarrow \Sigma_o^*$, DES isomorphism, \mathbf{G} -controllability, (\mathbf{G}, P) -observability, and the supremal quasi-congruence relation. Simply stated, natural projection P on a string $s \in \Sigma^*$ erases all the occurrences of $\sigma \in \Sigma$ in s such that $\sigma \notin \Sigma_o$, namely $P\sigma = \varepsilon$ (the empty string); P is implemented as $\text{Project}(\mathbf{G}, \text{Image}/\text{Null}[])$, which returns a (state-minimal) DES \mathbf{PG} over Σ_o such that $L_m(\mathbf{PG}) = PL_m(\mathbf{G})$ and $L(\mathbf{PG}) = PL(\mathbf{G})$. Two DES are isomorphic if they are identical up to relabeling of states; \mathbf{G} -controllability is

² We may safely assume that the implementation *Sync* of synchronous product is always associative and commutative; for more on this technicality see ([14], Sect. 3.3).

the property required for a sublanguage of $L_m(\mathbf{G})$ to be synthesizable by a supervisory controller; observability says that if two strings ‘look the same’ (have the same projection under P), then a control decision rule that applies to one can be used for the other; while projection modulo supremal quasi-congruence produces a (possibly nondeterministic) abstraction (reduced version) of a DES \mathbf{G} , denoted $Supqc(\mathbf{G}, Image/Null[])$, which preserves observable transitions and the ‘observer’ property [21, 22]. As detailed in [14] these operations are available in a software implementation [23] and will be referred to here as needed.

2.2 Distributed Control without Communication Delay

Next we summarize the distributed control theory (assuming zero communication delay) reported in [5, 6]. First suppose \mathbf{G} is to be controlled to satisfy a specification language $L_m(\mathbf{SPEC}) \subseteq \Sigma^*$ represented by a DES \mathbf{SPEC} . Denote by $K \subseteq \Sigma^*$ the supremal controllable sublanguage of $L_m(\mathbf{G}) \cap L_m(\mathbf{SPEC})$ (for details see [14]). Assume K is represented by the DES \mathbf{SUP} , which can be computed (e.g. in the software [23]) as

$$\mathbf{SUP} = Supcon(\mathbf{G}, \mathbf{SPEC}); \quad (1)$$

thus \mathbf{SUP} has marked behavior $L_m(\mathbf{SUP}) = K$ and closed behavior $L(\mathbf{SUP}) = \overline{K}$.

Since $\mathbf{G} = Sync(\mathbf{G}_1, \dots, \mathbf{G}_n)$ is the synchronous product of independent components we seek to implement \mathbf{SUP} in distributed fashion by ‘localizing’ \mathbf{SUP} to each \mathbf{G}_i as proposed in [5, 6]. For this we bring in a family of local controllers $\mathbf{LOC} = \{\mathbf{LOC}_i | i \in N\}$, one for each \mathbf{G}_i , and define $L(\mathbf{LOC}) = \|\{L(\mathbf{LOC}_i) | i \in N\}$ and $L_m(\mathbf{LOC}) = \|\{L_m(\mathbf{LOC}_i) | i \in N\}$. \mathbf{LOC} can be computed (see [23]) as $\mathbf{LOC} = Localize(\mathbf{G}, \mathbf{SUP})$; and it is shown in [5, 6] that

$$L(\mathbf{G}) \cap L(\mathbf{LOC}) = L(\mathbf{SUP}) \quad (2a)$$

$$L_m(\mathbf{G}) \cap L_m(\mathbf{LOC}) = L_m(\mathbf{SUP}) \quad (2b)$$

Generally, each local controller has a much smaller state set than \mathbf{SUP} and a smaller event subset of Σ , containing just the events of its corresponding plant component, together with those (‘communication’) events from other components that are essential to make correct control decisions.

3 Distributed Control with Communication Delay

Cai and Wonham [5] discuss a boundary case of optimal distributed control that is *fully-localizable* where inter-agent communication is not needed, namely the alphabet of each local controller \mathbf{LOC}_i is simply Σ_i , so that \mathbf{LOC}_i observes only events in its own agent \mathbf{G}_i . In this case, no issue of delay will arise. The more general and usual case is that inter-agent communication is imperative.

For simplicity assume temporarily that the system \mathbf{G} consists of two components \mathbf{G}_1 and \mathbf{G}_2 , and let the monolithic supervisor \mathbf{SUP} (in (1)) be given. Using

Localize compute local controllers \mathbf{LOC}_1 with event set $\Sigma_{\mathbf{LOC}_1}$ and \mathbf{LOC}_2 with event set $\Sigma_{\mathbf{LOC}_2}$, and then the local controlled behaviors

$$\mathbf{SUP}_1 = \text{Sync}(\mathbf{G}_1, \mathbf{LOC}_1) \quad (3)$$

$$\mathbf{SUP}_2 = \text{Sync}(\mathbf{G}_2, \mathbf{LOC}_2). \quad (4)$$

By the localization theory of [5, 6] we know that their synchronized behavior, say $\mathbf{LOCSUP} = \text{Sync}(\mathbf{SUP}_1, \mathbf{SUP}_2)$, will agree with that of the monolithic control \mathbf{SUP} (in (1)), namely

$$\text{Isomorph}(\mathbf{LOCSUP}, \mathbf{SUP}) = \text{true}. \quad (5)$$

In the general localization theory (instantaneous) inter-agent communication is both possible and necessary, so the alphabet $\Sigma_{\mathbf{LOC}_1}$ of \mathbf{LOC}_1 (resp. \mathbf{LOC}_2) will include elements (*communication events*) from Σ_2 (resp. Σ_1) as well as events from its ‘private’ alphabet Σ_1 (resp. Σ_2). Let $\Sigma_{com,1}$ (resp. $\Sigma_{com,2}$) represent the set of communication events from Σ_2 (resp. Σ_1), i.e. $\Sigma_{com,1} = \Sigma_{\mathbf{LOC}_1} - \Sigma_1$ (resp. $\Sigma_{com,2} = \Sigma_{\mathbf{LOC}_2} - \Sigma_2$); then the set of communication events in \mathbf{LOCSUP} (i.e. \mathbf{SUP}) is

$$\Sigma_{com} = \Sigma_{com,1} \cup \Sigma_{com,2}. \quad (6)$$

By (3) and (4), the alphabet $\Sigma_{\mathbf{SUP}_1}$ of \mathbf{SUP}_1 is

$$\Sigma_{\mathbf{SUP}_1} = \Sigma_1 \cup \Sigma_{com,1}, \quad (7)$$

and the alphabet $\Sigma_{\mathbf{SUP}_2}$ of \mathbf{SUP}_2 is

$$\Sigma_{\mathbf{SUP}_2} = \Sigma_2 \cup \Sigma_{com,2}. \quad (8)$$

We say that a communication event in $\Sigma_{com,1}$ is *imported* from \mathbf{G}_2 by \mathbf{LOC}_1 (resp. $\Sigma_{com,2}$, \mathbf{G}_1 and \mathbf{LOC}_2).

Next we model the way selected communication events are imported with indefinite time delay and call such events *channeled events*. Let Σ_{chn} represent the set of channeled events; then $\Sigma_{chn} \subseteq \Sigma_{com}$ (Σ_{com} is defined in (6)). For example assume that communication event r in Σ_2 is transmitted to \mathbf{LOC}_1 from \mathbf{G}_2 via a channel modeled as the (2-state) DES $\mathbf{C2r1}$ in Fig. 1³; then r is a channeled event. In the transition structure of \mathbf{LOC}_1 , hence also of \mathbf{SUP}_1 , we replace every instance of event r with a new event r' , the ‘output’ of $\mathbf{C2r1}$ corresponding to input r (we call r' the *signal event* of r); call these modified models \mathbf{LOC}'_1 , \mathbf{SUP}'_1 . Thus if and when r happens to occur (in \mathbf{G}_2) $\mathbf{C2r1}$ is driven by synchronization from its initial state 0 into state 1; on the eventual (and spontaneous) execution of event r' , which resets $\mathbf{C2r1}$ to state 0, the execution of r' will be forced by synchronization in \mathbf{LOC}'_1 , hence in \mathbf{SUP}'_1 ; of course each of the latter DES must be in an appropriate state for such synchronization to occur, namely exactly a state in which r was enabled originally in \mathbf{LOC}_1 and \mathbf{SUP}_1 . In the standard untimed model of DES employed here, the ‘time delay’ between an occurrence of r and

³ Communications among local supervisors can be modeled in different ways, e.g. [7–11]. Although in the 2-state automaton model, an event cannot be transmitted unless its channel is available, we select this model because it has simpler structure (2 states and 2 transitions), transmits events separately, and leads to a positive result (with this model, the local supervisors with communication delay achieve optimal and nonblocking supervision under suitable conditions).

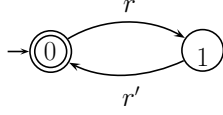


Fig. 1 Communication Channel **C2r1** (In the transition diagram of a DES, we use the circle with \rightarrow to represent the initial state and double circle to represent the marker state.)

r' is unspecified and can be considered unbounded; indeed, nothing in our model so far implies that r' necessarily ever occurs at all because, subsequent to the occurrence of r in \mathbf{G}_2 , \mathbf{SUP}'_1 might conceivably move to states (by events other than r') where r' can never be executed. As a convention, the control status of r' (controllable or uncontrollable) is taken to be that of r . Suppose in particular that r in Σ_2 is controllable. Since \mathbf{LOC}_1 has ‘control authority’ only over controllable events in its private alphabet Σ_1 , \mathbf{LOC}'_1 never attempts to disable r' directly; r' can only be disabled implicitly by the ‘upstream’ disablement by \mathbf{LOC}_2 of r .

In general \mathbf{LOC}'_1 ‘knows’ that r has occurred in \mathbf{G}_2 only when it receives r' ; meanwhile, other events may have occurred in \mathbf{G}_2 . The only constraint placed on events in \mathbf{G}_2 is that r cannot occur again until r' has finally reset **C2r1** and the communication cycle is ready to repeat⁴. In other words, event r will be delayed in re-occurring until the channel used to transmit event r is again available. If event r is controllable, it can be disabled or delayed by the local controller \mathbf{LOC}_2 ; but if event r is uncontrollable, the constraint placed on \mathbf{G}_2 will require that r' should reset **C2r1** before r can occur again, possibly in violation of the intended meaning of ‘uncontrollable’. This issue will be discussed in Sect. 3.3. The channel **C2r1** is not considered a control device, but rather an intrinsic component of the physical system being modeled; it will be ‘hard-wired’ into the model by synchronous product with \mathbf{G}_1 and \mathbf{G}_2 .

Continuing with this special case we consider the joint behavior of \mathbf{G}_1 , \mathbf{G}_2 and **C2r1** under control of \mathbf{LOC}'_1 and \mathbf{LOC}_2 , namely

$$\begin{aligned} \mathbf{SUP}' &:= \text{Sync}(\mathbf{G}_1, \mathbf{LOC}'_1, \mathbf{C2r1}, \mathbf{G}_2, \mathbf{LOC}_2) \\ &= \text{Sync}(\mathbf{SUP}'_1, \mathbf{C2r1}, \mathbf{SUP}_2) \end{aligned} \quad (9)$$

defined over the alphabet $\Sigma_1 \cup \{r'\} \cup \Sigma_2$. We refer to \mathbf{SUP}' as the *channeled behavior* of \mathbf{SUP} (in (1)) with r being the channeled event (i.e. $\Sigma_{chn} = \{r\}$).

3.1 Delay-robustness and Delay-criticality

In this subsection, we formalize the definition and present an effective computational test for delay-robustness.

Of principal interest is whether or not the communication delay between successive occurrences of r and r' is tolerable in the intuitive sense indicated above.

⁴ In a more fine-grained model we could set $r' = r'_{21}r'_{12}$ where r'_{21} signals to \mathbf{LOC}'_1 the occurrence of r in \mathbf{G}_2 , while r'_{12} represents an acknowledgement to \mathbf{LOC}_2 that r'_{21} has occurred in \mathbf{SUP}'_1 . Here the accumulated delay is assumed to be captured by **C2r1** modeled just with r' .

Let Σ_{sig} be the set of new events introduced by the communication channel, in which each element is the signal event of an event in Σ_{chn} , i.e.

$$\Sigma_{sig} = \{\sigma' | \sigma \in \Sigma_{chn}, \sigma' \text{ is the signal event of } \sigma\}. \quad (10)$$

In **SUP'** (in (9)), $\Sigma_{chn} = \{r\}$ and $\Sigma_{sig} = \{r'\}$. Then the event set of **SUP'** will be $\Sigma' = \Sigma \cup \Sigma_{sig} = \Sigma \cup \{r'\}$. Let $P : \Sigma'^* \rightarrow \Sigma^*$ be the natural projection of Σ'^* onto Σ^* [14], i.e. P maps r' to ϵ (empty string).

To define whether or not **SUP'** with alphabet Σ' has the same behavior as **SUP**, when viewed through P , we require that

1. anything **SUP** can do is the P -projection of something **SUP'** can do (**SUP'** is ‘complete’); and

2. no P -projection of anything **SUP'** can do is disallowed by **SUP** (**SUP'** is ‘correct’).

For completeness we need at least the inclusions

$$PL(\mathbf{SUP}') \supseteq L(\mathbf{SUP}) \quad (11)$$

$$PL_m(\mathbf{SUP}') \supseteq L_m(\mathbf{SUP}) \quad (12)$$

In addition, however, we need the following *observer property* of P with respect to **SUP'** and **SUP**. Suppose **SUP'** executes string $s \in L(\mathbf{SUP}')$, which will be viewed as $Ps \in L(\mathbf{SUP})$. As **SUP** is nonblocking, there exists $w \in \Sigma^*$ such that $(Ps)w \in L_m(\mathbf{SUP})$. For any such w ‘chosen’ by **SUP**, completeness should require the ability of **SUP'** to provide a string $v \in \Sigma'^*$ with the property $Pv = w$ and $sv \in L_m(\mathbf{SUP}')$. Succinctly (cf. [14, 22])

$$\begin{aligned} (\forall s \in \Sigma'^*)(\forall w \in \Sigma^*) \quad & s \in L(\mathbf{SUP}') \ \& \ (Ps)w \in L_m(\mathbf{SUP}) \\ \Rightarrow (\exists v \in \Sigma'^*) \quad & Pv = w \ \& \ sv \in L_m(\mathbf{SUP}'). \end{aligned} \quad (13)$$

Remark 1 In ([14], Chapt. 6), P is defined to be an $L_m(\mathbf{SUP}')$ -observer if

$$\begin{aligned} (\forall s \in \Sigma'^*)(\forall w \in \Sigma^*) \quad & s \in L(\mathbf{SUP}') \ \& \ (Ps)w \in PL_m(\mathbf{SUP}') \\ \Rightarrow (\exists v \in \Sigma'^*) \quad & Pv = w \ \& \ sv \in L_m(\mathbf{SUP}'). \end{aligned}$$

It is clear that when $PL_m(\mathbf{SUP}') = L_m(\mathbf{SUP})$, the observer property of P with respect to **SUP'** and **SUP** is identical with the $L_m(\mathbf{SUP}')$ -observer property of P .

Briefly, we define **SUP'** to be *complete* relative to **SUP** if (11), (12) and (13) hold.

Dually, but more simply, we say that **SUP'** is *correct* relative to **SUP** if

$$PL(\mathbf{SUP}') \subseteq L(\mathbf{SUP}) \quad (14)$$

$$PL_m(\mathbf{SUP}') \subseteq L_m(\mathbf{SUP}) \quad (15)$$

To summarize, we make

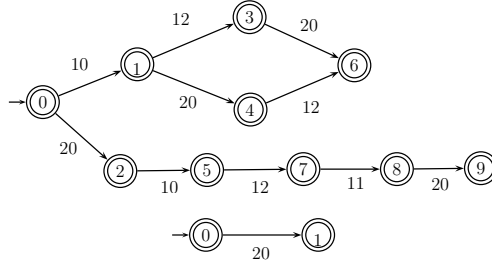


Fig. 2 Example 1: \mathbf{SUP}_1 and \mathbf{SUP}_2

Definition 1 For given \mathbf{SUP}' in (9) and $\Sigma_{chn} = \{r\}$, \mathbf{SUP} (in (1)) is *delay-robust* relative to Σ_{chn} provided \mathbf{SUP}' is complete and correct relative to \mathbf{SUP} , namely, formulas (11)-(15) hold, or explicitly

$$PL(\mathbf{SUP}') = L(\mathbf{SUP}) \quad (16)$$

$$PL_m(\mathbf{SUP}') = L_m(\mathbf{SUP}) \quad (17)$$

$$P \text{ has the observer property (13) with respect to } \mathbf{SUP}' \text{ and } \mathbf{SUP}. \quad (13\text{bis})$$

The following example shows why the observer property is really needed; for if (16) and (17) hold, but (13) fails, \mathbf{SUP}' may have behavior distinguishable from that of \mathbf{SUP} .

Example 1 Let \mathbf{SUP}_1 and \mathbf{SUP}_2 be the generators shown in Fig. 2; assume event 20 in \mathbf{SUP}_2 is exported to \mathbf{SUP}_1 , i.e., $r = 20$ and $r' = 120$; \mathbf{SUP}'_1 is obtained by replacing 20 in \mathbf{SUP}_1 by 120, and \mathbf{SUP}' is obtained by (9). By inspection of Fig. 3, (16) and (17) are verified to hold. However, we can see that (13) fails. Let $s = 20.10.120.12 \in L(\mathbf{SUP}')$; then $Ps = 20.10.12$. Now $(Ps).11 = 20.10.12.11 \in L_m(\mathbf{SUP})$; but there does not exist a string v such that $Pv = 11$ and $sv \in L_m(\mathbf{SUP}')$. Thus, \mathbf{SUP} can execute 11 after Ps , but \mathbf{SUP}' cannot execute 11 after s . This shows that \mathbf{SUP}' has behavior distinct from that of \mathbf{SUP} .

Since \mathbf{SUP} is a nonblocking supervisor, delay-robustness of \mathbf{SUP} also requires that \mathbf{SUP}' should be nonblocking, i.e.

$$\overline{L_m(\mathbf{SUP}')} = L(\mathbf{SUP}'), \quad (18)$$

as can easily be derived from (13), (16) and (17). The following example shows that if delay-robustness fails, then transmission delay of r can lead to blocking in \mathbf{SUP}' .

Example 2 Let \mathbf{SUP}_1 and \mathbf{SUP}_2 be the generators shown in Fig. 4, and assume event 22 in \mathbf{SUP}_2 is exported to \mathbf{SUP}_1 , i.e., $r = 22$ and $r' = 122$; \mathbf{SUP}'_1 is obtained by replacing 22 in \mathbf{SUP}_1 by 122. Then \mathbf{SUP} is nonblocking, but \mathbf{SUP}' obtained by (9) is blocking, as shown in Fig. 5. To see why, start from the initial state, and suppose event 22 has occurred in \mathbf{SUP}_2 but that \mathbf{SUP}'_1 has not received the corresponding event 122. Then \mathbf{SUP}'_1 may execute event 13, which is immediately observed by \mathbf{SUP}_2 ; however, if 13 occurs, \mathbf{SUP}_1 and \mathbf{SUP}_2 cannot accomplish their task synchronously; hence the system blocks.

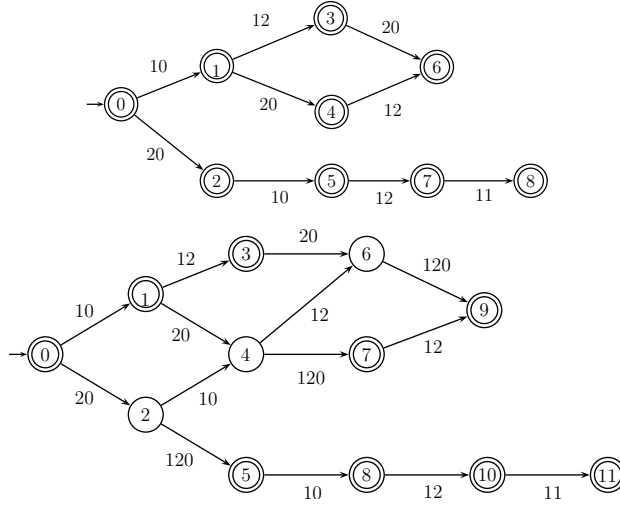


Fig. 3 Example 1: **SUP** and **SUP'**

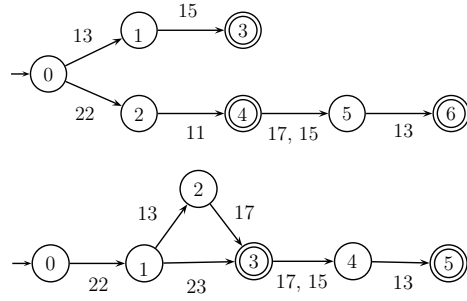


Fig. 4 Example 2: **SUP₁** and **SUP₂**

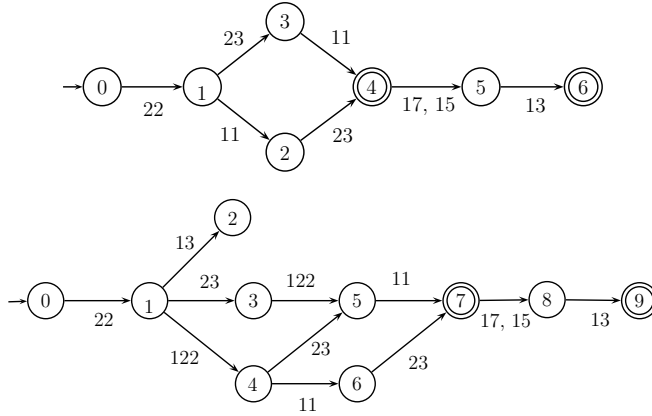


Fig. 5 Example 2: **SUP** and **SUP'**

Given **SUP**, Σ_{chn} , Σ_{sig} and **SUP'**, we wish to verify whether or not **SUP** is delay-robust relative to Σ_{chn} . For this we need the concept of “supremal quasi-

congruence” [14,21] and the operator $Supqc$ [14] which projects a given **DES** over the alphabet Σ' to **QCDES**, the corresponding quotient **DES** over $\Sigma^* = P(\Sigma'^*)$. We denote the counterpart computing procedure by

$$\mathbf{QCDES} = Supqc(\mathbf{DES}, Null[P])$$

where $Null[P]$ is the event subset $\Sigma' - \Sigma$ that P maps to the empty string ϵ ; for details see [14]⁵. Let $\mathbf{QCDES} = (Z, \Sigma, \zeta, z_0, Z_m)$. In general **QCDES** will be nondeterministic and include silent (ϵ -) transitions. If no silent or nondeterministic transitions happen to appear in **QCDES**, the latter is said to be ‘structurally deterministic’. Formally, **QCDES** is *structurally deterministic* if, for all $z \in Z$ and $s \in \Sigma^*$, we have

$$\zeta(z, s) \neq \emptyset \Rightarrow |\zeta(z, s)| = 1.$$

It is known that structural determinism of **QCDES** is equivalent to the condition that P is an $L_m(\mathbf{DES})$ -observer (cf. ([14], Theorem 6.7.1)).

Given deterministic generators **A** and **B** over the same alphabet, we write $\mathbf{A} \subseteq \mathbf{B}$ iff $L_m(\mathbf{A}) \subseteq L_m(\mathbf{B})$ and $L(\mathbf{A}) \subseteq L(\mathbf{B})$; and $\mathbf{A} \approx \mathbf{B}$ to mean both $(\mathbf{A} \subseteq \mathbf{B})$ and $(\mathbf{B} \subseteq \mathbf{A})$. Using standard computing tools (e.g. [14]) for DES isomorphism and minimal (Nerode) state reduction we have that $\mathbf{A} \approx \mathbf{B}$ iff

$$Isomorph(Minstate(\mathbf{A}), Minstate(\mathbf{B})) = true.$$

Clearly, “ \approx ” is transitive, i.e., for given **A**, **B** and **C**,

$$\mathbf{A} \approx \mathbf{B} \ \& \ \mathbf{B} \approx \mathbf{C} \Rightarrow \mathbf{A} \approx \mathbf{C}.$$

Now let $\mathbf{SUP} = (X, \Sigma, \xi, x_0, X_m)$ (in (1)), $\mathbf{SUP}' = (Y, \Sigma', \eta, y_0, Y_m)$ (in (9)). Let

$$\mathbf{QCSUP}' = Supqc(\mathbf{SUP}', Null[r'])$$

where $Supqc$ maps r' to ϵ and write $\mathbf{QCSUP}' = (\bar{Y}, \Sigma, \bar{\eta}, \bar{y}_0, \bar{Y}_m)$.

The following theorem provides an effective test for whether or not the communication delay is tolerable, i.e., **SUP** is delay-robust.

Theorem 1 ***SUP** is delay-robust relative to $\Sigma_{chn} (= \{r\})$ if and only if \mathbf{QCSUP}' is structurally deterministic, and isomorphic to **SUP**.*

As indicated above, \mathbf{QCSUP}' can be computed by $Supqc$ and isomorphism of DES can be verified by $Isomorph$. Hence, Theorem 1 provides an effective computational criterion for delay-robustness. Before Theorem 1 is proved, a special relation between \mathbf{QCSUP}' and \mathbf{PSUP}' must be established; a proof is in Appendix A.

Proposition 1 *If \mathbf{QCSUP}' is structurally deterministic, then it is a canonical (minimal-state) generator for $PL_m(\mathbf{SUP}')$.*

⁵ This procedure can also be phrased in terms of ‘bisimulation equivalence’ [24], as explained in [21].

Proof of Theorem 1. (If) From Proposition 1, \mathbf{QCSUP}' is a minimal state generator of $PL_m(\mathbf{SUP}')$. So, $\mathbf{QCSUP}' \approx \mathbf{PSUP}'$. As \mathbf{QCSUP}' is isomorphic to \mathbf{SUP} , $\mathbf{QCSUP}' \approx \mathbf{SUP}$. Hence, $\mathbf{SUP} \approx \mathbf{PSUP}'$, i.e. formulas (16) and (17) both hold. For (13), since \mathbf{QCSUP}' is structurally deterministic ([14], Theorem 6.7.1), P is an $L_m(\mathbf{SUP}')$ -observer; by Remark 1 and (17), P has the observer property with respect to \mathbf{SUP}' and \mathbf{SUP} . Thus by Definition 1, \mathbf{SUP} is delay-robust relative to Σ_{chn} .

(Only if) By Remark 1, condition (13) and formula (17) imply that P is an $L_m(\mathbf{SUP}')$ -observer; thus \mathbf{QCSUP}' is deterministic ([14]). By Proposition 1, $\mathbf{QCSUP}' \approx \mathbf{PSUP}'$. Formulas (16) and (17) say that $\mathbf{PSUP}' \approx \mathbf{SUP}$. Hence $\mathbf{QCSUP}' \approx \mathbf{SUP}$. Finally, we conclude that \mathbf{QCSUP}' is isomorphic to \mathbf{SUP} . \square

We have now obtained an effective tool to determine whether or not \mathbf{SUP} is delay-robust relative to $\Sigma_{chn} = \{r\}$. If \mathbf{SUP} is not delay-robust relative to r , we say that r is *delay-critical* for \mathbf{SUP} . In that case, communication of r (with delay, as r') could result in violation of a specification. If r is delay-critical, and if such violation is inadmissible, then r must be transmitted instantaneously to the agent (in this case, \mathbf{LOC}_1) that imports it – where “instantaneous” must be quantified on the application-determined time scale.

3.2 Delay-robustness for Multiple Events

In this subsection, we consider delay-robustness for multiple events. First, we adopt the result of Theorem 1 as the basis of a new definition and extend delay-robustness naturally to multiple events. Then we prove that delay-robustness of a set R_2 (of multiple events) implies that delay-robustness holds for any subset of R_2 .

Definition 2 Let $R_2 \subseteq \Sigma_2$ be a subset of events r imported from \mathbf{G}_2 by \mathbf{LOC}_1 via their corresponding channels $\mathbf{C2r1}$ (i.e. $\Sigma_{chn} = R_2$), and let \mathbf{SUP}_1 be modified to \mathbf{SUP}'_1 by replacing each r by its transmitted version r' as before. Let

$$\mathbf{SUP}' := \text{Sync}(\mathbf{SUP}'_1, \{\mathbf{C2r1} | r \in R_2\}, \mathbf{SUP}_2).$$

Then \mathbf{SUP} is *delay-robust relative to the event subset R_2* provided

$$\text{Isomorph}(\text{Supqc}(\mathbf{SUP}', \text{Null}[\{r' | r \in R_2\}]), \mathbf{SUP}) = \text{true}.$$

Note that the property of \mathbf{SUP} described in Definition 2 is stricter than in Definition 1: that \mathbf{SUP} is delay-robust with respect to each event $r \in R_2$ taken separately does not imply that \mathbf{SUP} is delay-robust with respect to R_2 as a subset; however, that \mathbf{SUP} is delay-robust with respect to R_2 does imply that \mathbf{SUP} is delay-robust with respect to each separate event $r \in R_2$. The former statement will be illustrated in the **WORKCELL** example in Sect. 4. We confirm the latter as follows.

Theorem 2 Let $R_2 = \{\alpha_1, \dots, \alpha_n\}$ and let $A_i = \{\alpha_1, \dots, \alpha_i\}$ ($1 \leq i \leq n$). If \mathbf{SUP} is delay-robust with respect to A_n ($= R_2$), then \mathbf{SUP} is delay-robust with respect to A_{n-1} ($= A_n - \{\alpha_n\}$).

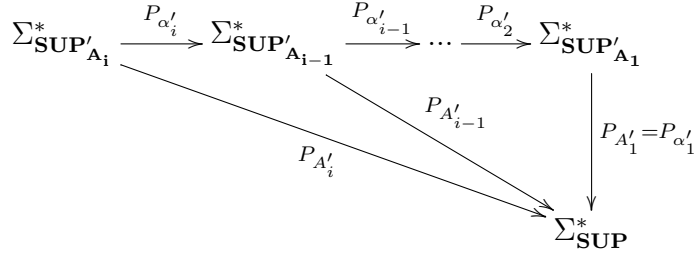


Fig. 6 $P_{A'_i}$ and $P_{\alpha'_i}$

By Theorem 2, if **SUP** is delay-robust with respect to R_2 , then it is delay-robust with respect to the subset $A_n - \{\alpha_i\}$ for arbitrary α_i . Applying Theorem 2 on subsets of R_2 , we conclude that

Corollary 1 *If **SUP** is delay-robust with respect to R_2 , then it is delay-robust with respect to any subset of R_2 .*

Let \mathbf{SUP}'_{A_i} be the channeled behavior of **SUP** with exactly the events in A_i being channeled events (A'_i is the event set of signal events corresponding to events in A_i), i.e.

$$\mathbf{SUP}'_{A_i} = \text{Sync}(\mathbf{SUP}'_1, \{\mathbf{CH}_\alpha | \alpha \in A_i\}, \mathbf{SUP}'_2) \quad (19)$$

where \mathbf{SUP}'_k ($k = 1, 2$) is obtained by replacing α in \mathbf{SUP}_k by α' (α' is the signal event of α) and \mathbf{CH}_α is the communication channel for α . $\Sigma_{\mathbf{SUP}'_{A_i}} = \Sigma_{\mathbf{SUP}} \cup A'_i$ is the event set of \mathbf{SUP}'_{A_i} .

Remark 2 If $\alpha_{i+1} \in \Sigma_{\mathbf{SUP}_1}$, then

$$\begin{aligned} \mathbf{SUP}'_{A_{i+1}} &= \text{Sync}(\mathbf{SUP}''_1, \{\mathbf{CH}_\alpha | \alpha \in A_{i+1}\}, \mathbf{SUP}'_2) \\ &= \text{Sync}(\mathbf{SUP}''_1, \mathbf{CH}_{\alpha_{i+1}}, \text{Sync}(\{\mathbf{CH}_\alpha | \alpha \in A_i\}, \mathbf{SUP}'_2)) \end{aligned}$$

where \mathbf{SUP}''_1 is obtained by replacing α_{i+1} in \mathbf{SUP}'_1 by α'_{i+1} . So $\mathbf{SUP}'_{A_{i+1}}$ can be treated as the channeled behavior of \mathbf{SUP}'_{A_i} with α_{i+1} being the only channeled event and $\mathbf{CH}_{\alpha_{i+1}}$ its communication channel.

Define natural projections $P_{A'_i} : \Sigma_{\mathbf{SUP}'_{A_i}}^* \rightarrow \Sigma_{\mathbf{SUP}}^*$ which maps $\sigma' \in A'_i$ to ϵ (empty string) and $P_{\alpha'_i} : \Sigma_{\mathbf{SUP}'_{A_i}}^* \rightarrow \Sigma_{\mathbf{SUP}'_{A_{i-1}}}^*$ which maps α'_i to ϵ ; then we have

$$P_{A'_i} = P_{A'_{i-1}} P_{\alpha'_i} = \dots = P_{\alpha'_1} (\dots (P_{\alpha'_{i-1}} P_{\alpha'_i})) = P_{\alpha'_1} \dots P_{\alpha'_{i-1}} P_{\alpha'_i}, \quad (20)$$

as shown in Fig. 6.

By definition of delay-robust (Definition 1), we must verify (13), (16) and (17) for **SUP**, $\mathbf{SUP}'_{A_{n-1}}$ and $P_{A'_{n-1}}$, i.e.

$$L(\mathbf{SUP}) = P_{A'_{n-1}} L(\mathbf{SUP}'_{A_{n-1}}), \quad (21)$$

$$L_m(\mathbf{SUP}) = P_{A'_{n-1}} L_m(\mathbf{SUP}'_{A_{n-1}}), \quad (22)$$

$$P_{A'_{n-1}} \text{ has the observer property with respect to } \mathbf{SUP} \text{ and } \mathbf{SUP}'_{A_{n-1}}. \quad (23)$$

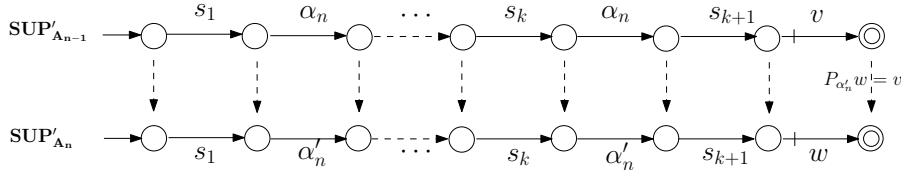


Fig. 7 $sv \notin L_m(\text{SUP}'_{A_{n-1}}) \Rightarrow tw \notin L_m(\text{SUP}'_{A_n})$ (An arrow with a vertical bar represents that the corresponding transition with a string or an event is not defined.)

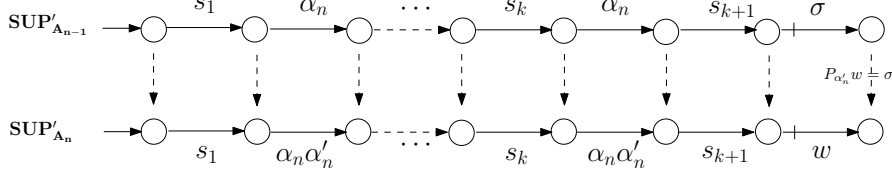


Fig. 8 $s\sigma \notin L(\text{SUP}'_{A_{n-1}}) \Rightarrow tw \notin L(\text{SUP}'_{A_n})$

Equations (21) and (22) can be proved from the process of obtaining $\text{SUP}'_{A_{n-1}}$ and SUP'_{A_n} , and the relationships among $P_{A'_{n-1}}$, $P_{\alpha'_n}$ and $P_{A'_n}$ shown in Fig. 6. Condition (23) cannot be obtained from the observer property of $P_{A'_n}$ (with respect to SUP and SUP'_{A_n}), since SUP'_{A_n} has more complex structure than $\text{SUP}'_{A_{n-1}}$, inasmuch as the component SUP'_1 (suppose α_n is imported by SUP'_1 from \mathbf{G}_2) in SUP'_{A_n} receives the occurrence of α_n (i.e. α'_n) with indefinite delay, but $P_{A'_n}$ has the same image (observable event set) as $P_{A'_{n-1}}$, as shown in Fig. 6. So, we prove (23) by contraposition as follows: a ‘bad’ string in $L(\text{SUP}'_{A_{n-1}})$ (causing $P_{A'_{n-1}}$ to fail to have the observer property (23)) implies the existence of a ‘bad’ string in $L(\text{SUP}'_{A_n})$ (causing $P_{A'_n}$ to fail to have the observer property). For this, we need the following lemmas, proved in Appendix B. Lemma 1 shows the relationship between $\text{SUP}'_{A_{i-1}}$ and SUP'_{A_i} . Lemmas 2 and 3 show that if a string $s \in L(\text{SUP}'_{A_{n-1}})$ causes $P_{A'_{n-1}}$ to fail to have the observer property, then there exists $t \in L(\text{SUP}'_{A_n})$ corresponding to s showing that $P_{A'_n}$ fails to have the observer property: as described in Fig. 7, Lemma 2 says that if in $\text{SUP}'_{A_{n-1}}$ there is a string s which cannot lead to a marker state by $v \in \Sigma_{\text{SUP}'_{A_{n-1}}}^*$, then in SUP'_{A_n} there exists a string t corresponding to s such that if $P_{\alpha'_n}w = v$, then t cannot arrive to a marker state through w ; as in Fig. 8, Lemma 3 says that if in $\text{SUP}'_{A_{n-1}}$ there is a string s after which $\sigma \in \Sigma_{\text{SUP}'_{A_{n-1}}}$ is not defined, then in $\text{SUP}'_{A_{n-1}}$ there exists a string t corresponding to s such that if $P_{\alpha'_n}w = \sigma$, then w is not defined after t .

Lemma 1 Let $\text{SUP}_{A'_0}$ represent SUP . For $1 \leq i \leq n$,

$$\begin{aligned} L(\text{SUP}'_{A_{i-1}}) &\subseteq P_{\alpha'_i} L(\text{SUP}'_{A_i}) \\ L_m(\text{SUP}'_{A_{i-1}}) &\subseteq P_{\alpha'_i} L_m(\text{SUP}'_{A_i}) \end{aligned}$$

Lemma 2 Suppose there exist $s \in L(\mathbf{SUP}'_{\mathbf{A}_{n-1}})$ and $v \in \Sigma_{\mathbf{SUP}'_{\mathbf{A}_{n-1}}}^*$ such that $\# \alpha_n(v) = 0$ and $sv \notin L_m(\mathbf{SUP}'_{\mathbf{A}_{n-1}})$. Then there exists $t \in L(\mathbf{SUP}'_{\mathbf{A}_n})$ such that $P_{\alpha'_n} t = s$ and for any $w \in \Sigma_{\mathbf{SUP}'_{\mathbf{A}_n}}^*$ if $P_{\alpha'_n} w = v$, then $tw \notin L_m(\mathbf{SUP}'_{\mathbf{A}_n})$.

Lemma 3 Suppose there exist $s \in L(\mathbf{SUP}'_{\mathbf{A}_{n-1}})$ and $\sigma \in \Sigma_{\mathbf{SUP}'_{\mathbf{A}_{n-1}}}$ such that $s\sigma \notin L(\mathbf{SUP}'_{\mathbf{A}_{n-1}})$. Then there exists $t \in L(\mathbf{SUP}'_{\mathbf{A}_n})$ such that $P_{\alpha'_n} t = s$ and for any $w \in \Sigma_{\mathbf{SUP}'_{\mathbf{A}_n}}^*$ if $P_{\alpha'_n} w = \sigma$, then $tw \notin L(\mathbf{SUP}'_{\mathbf{A}_n})$.

Proof of Theorem 2. By definition of delay-robust, we must prove (21), (22) and (23).

(1) For (21), we first show that $L(\mathbf{SUP}) \subseteq P_{n-1}L(\mathbf{SUP}')$. By Lemma 1,

$$\begin{aligned} L(\mathbf{SUP}) &\subseteq P_{\alpha'_1} L(\mathbf{SUP}'_{\mathbf{A}_1}) \\ &\subseteq P_{\alpha'_1} (P_{\alpha'_2} L(\mathbf{SUP}'_{\mathbf{A}_2})) \\ &\subseteq P_{\alpha'_1} P_{\alpha'_2} \dots P_{\alpha'_{n-1}} L(\mathbf{SUP}'_{\mathbf{A}_{n-1}}) \\ &= P_{A'_{n-1}} L(\mathbf{SUP}'_{\mathbf{A}_{n-1}}) \quad (\text{by (20)}) \end{aligned}$$

Similarly, $L(\mathbf{SUP}'_{\mathbf{A}_{n-1}}) \subseteq P_{\alpha'_n} L(\mathbf{SUP}'_{\mathbf{A}_n})$. So,

$$\begin{aligned} P_{A'_{n-1}} L(\mathbf{SUP}'_{\mathbf{A}_{n-1}}) &\subseteq P_{A'_{n-1}} (P_{\alpha'_n} L(\mathbf{SUP}'_{\mathbf{A}_n})) \\ &= P_{A'_n} L(\mathbf{SUP}'_{\mathbf{A}_n}) \\ &\subseteq L(\mathbf{SUP}). \quad (\text{since } \mathbf{SUP} \text{ is delay-robust with} \\ &\quad \text{respect to } R_2 \text{ (i.e. } A_n)) \end{aligned}$$

Hence, $L(\mathbf{SUP}) = P_{A'_{n-1}} L(\mathbf{SUP}'_{\mathbf{A}_{n-1}})$.

(2) Equation (22) is proved similarly.

(3) For (23), we prove the contrapositive.

Assume that $P_{A'_{n-1}}$ does not have the observer property with respect to \mathbf{SUP} and $\mathbf{SUP}'_{\mathbf{A}_{n-1}}$, i.e. there exist $s \in L(\mathbf{SUP}'_{\mathbf{A}_{n-1}})$ and $w \in \Sigma^*$ such that $(P_{A'_{n-1}} s)w \in L_m(\mathbf{SUP})$ and for all $v \in \Sigma_{\mathbf{SUP}'_{\mathbf{A}_{n-1}}}^*$, either $P_{A'_{n-1}} v \neq w$ or $sv \notin L_m(\mathbf{SUP}'_{\mathbf{A}_{n-1}})$. Thus, for all $v \in \Sigma_{\mathbf{SUP}'_{\mathbf{A}_{n-1}}}^*$, if $P_{A'_{n-1}} v = w$, then $sv \notin L_m(\mathbf{SUP}'_{\mathbf{A}_{n-1}})$. Let $\# \alpha_n(v) = m$; we consider the following two cases.

(i) $m = 0$. By Lemma 2, there exists $t \in L(\mathbf{SUP}'_{\mathbf{A}_n})$ such that $P_{\alpha'_n}(t) = s$ and for any $u \in \Sigma_{\mathbf{SUP}'_{\mathbf{A}_n}}^*$ if $P_{\alpha'_n} u = v$, then $tu \notin L_m(\mathbf{SUP}'_{\mathbf{A}_n})$. However, $P_{A'_n}(tu) = P_{A_{n-1}'}(P_{\alpha'_n}(tu)) = P_{A_{n-1}'}(sv) = P_{A_{n-1}'}(s)w \in L_m(\mathbf{SUP})$. Hence, $P_{A'_n}$ does not have the observer property with respect to \mathbf{SUP} and $\mathbf{SUP}'_{\mathbf{A}_n}$.

(ii) $m > 0$. Write $v = v_1 \alpha_n v_2$ where $\# \alpha_n(v_2) = 0$. Let $s' = sv_1 \alpha_n$. If $s' \in L(\mathbf{SUP}'_{\mathbf{A}_{n-1}})$, by Lemma 2 there exists $t \in L(\mathbf{SUP}'_{\mathbf{A}_n})$ such that $P_{\alpha'_n}(t) = s'$ and for any $u \in \Sigma_{\mathbf{SUP}'_{\mathbf{A}_n}}^*$ if $P_{\alpha'_n} u = v_2$, then $tu \notin L_m(\mathbf{SUP}'_{\mathbf{A}_n})$. So in this case $P_{A'_n}$ does not have the observer property with respect to \mathbf{SUP} and $\mathbf{SUP}'_{\mathbf{A}_n}$. If $s' \notin L(\mathbf{SUP}'_{\mathbf{A}_{n-1}})$, then because $s \in L(\mathbf{SUP}'_{\mathbf{A}_{n-1}})$, there must exist $v_{11}, v_{12} \in \Sigma_{\mathbf{SUP}'_{\mathbf{A}_{n-1}}}^*$ and $\sigma \in \Sigma_{\mathbf{SUP}'_{\mathbf{A}_{n-1}}}$ such that $v_1 = v_{11} \sigma v_{12}$ and $sv_{11} \in L(\mathbf{SUP}'_{\mathbf{A}_{n-1}})$, but $sv_{11} \sigma \notin L(\mathbf{SUP}'_{\mathbf{A}_{n-1}})$. By Lemma 3, there exists $t \in L(\mathbf{SUP}'_{\mathbf{A}_n})$ such that

$P_{\alpha'_n}(t) = sv_{11}$ and for any $u \in \Sigma_{\mathbf{SUP}'_{A_n}}^*$ if $P_{\alpha'_n}u = \sigma$, then $tu \notin L(\mathbf{SUP}'_{A_n})$. So, $tu v_{12} \alpha_n v_2 \notin L_m(\mathbf{SUP}'_{A_n})$. However,

$$\begin{aligned} P_{A'_n}(tuv_{12}\alpha_n v_2) &= P_{A_{n-1}'}(P_{\alpha'_n}(tuv_{12}\alpha_n v_2)) \\ &= P_{A_{n-1}'}(sv_{11}\sigma v_{12}\alpha_n v_2) \\ &= P_{A_{n-1}'}(sv) \\ &= P_{A_{n-1}'}(s)w \in L_m(\mathbf{SUP}). \end{aligned}$$

Hence, $P_{A'_n}$ does not have the observer property with respect to \mathbf{SUP} and \mathbf{SUP}'_{A_n} .

In both cases, the results contradict the assumption that \mathbf{SUP} is delay-robust with respect to $R_2 (= A_n)$; hence (23) holds after all. We conclude that \mathbf{SUP} is delay-robust with respect to A_{n-1} , as claimed by Theorem 2. \square

Finally, we define delay-robustness with respect to both channeled communication of $\Sigma_{com,1}$ from \mathbf{G}_2 to \mathbf{LOC}_1 and channeled communication of $\Sigma_{com,1}$ from \mathbf{G}_1 to \mathbf{LOC}_2 , i.e. $\Sigma_{chn} = \Sigma_{com} = \Sigma_{com,1} \cup \Sigma_{com,2}$ (as in (6)). In obvious notation, let

$$\mathbf{SUP}' := \text{Sync}(\mathbf{SUP}'_1, \{\mathbf{C2r1} | r \in \Sigma_{com,1}\}, \{\mathbf{C1r2} | r \in \Sigma_{com,2}\}, \mathbf{SUP}'_2).$$

Definition 3 \mathbf{SUP} is *delay-robust for distributed control by localization* provided the projected channeled behavior

$$\text{Supqc}(\mathbf{SUP}', \text{Null}[\{r1' | r1 \in \Sigma_{com,1}\} \cup \{r2' | r2 \in \Sigma_{com,2}\}])$$

is isomorphic to \mathbf{SUP} .

Delay-robustness in the sense of Definition 3 represents an ideal which is too strong a property to be expected in most practical situations. We discuss this issue in the light of the **WORKCELL** example in Sect. 4.

We note in passing that all the above results can be extended to decentralized controllers; for details see Appendix C.

3.3 Bounded and Unbounded Delay

The foregoing discussion of delay robustness covers channeled events in general, regardless of their control status, and is adequate if all channeled events happen to be controllable. In the case of uncontrollable channeled events, however, we must additionally examine whether channel delay introduces undesirable side effects or violates the conventional modeling assumption that uncontrollable events may occur spontaneously at states where they are enabled and should not be subject to external disablement. To this end, in the rest of this section we consider the re-occurrence of an uncontrollable channeled event, and then distinguish delay-robustness as bounded or unbounded according to whether such an event is blocked or not by its communication channel.

In our model the transmission of r from \mathbf{G}_2 to \mathbf{LOC}_1 is completed (by event r') with indefinite (unbounded) delay. A constraint imposed on \mathbf{SUP}' by the channel $\mathbf{C2r1}$ is that r cannot occur again until r' has reset $\mathbf{C2r1}$ and the communication cycle is ready to repeat. If r is controllable its re-occurrence can be disabled and

hence delayed until after the occurrence of r' corresponding to the previous occurrence of r . If, however, r is uncontrollable, then once it is re-enabled (by entrance of \mathbf{SUP}_2 to a state where r is defined) its re-occurrence cannot be externally delayed, according to the usual modeling assumption on uncontrollable events. In this sense the introduction of $\mathbf{C2r1}$ could conceivably conflict with the intention of the original DES model. To address this issue we examine (1) whether the (unpreventable) occurrence of an uncontrollable channeled event might violate the problem specification, and (2) whether communication delay of an uncontrollable event might violate a modeling assumption.

We refer to case (1) as an *implementation fault* (or simply *fault*), namely an uncontrollable channeled event r is executed by \mathbf{SUP}' despite the fact that, after the last previous occurrence of r , its channel is still waiting at state 1 for the reset event r' . Assume that \mathbf{SUP} is delay-robust with respect to r . It will be shown that a fault will not cause violation of the specification, namely the fault is accepted by (i.e. defined in) \mathbf{SUP} .

Let the DES

$$\mathbf{C2r1} = (\{0, 1\}, \{r, r'\}, \delta, 0, \{0\}); \quad (24)$$

below for $\mathbf{C2r1}$ we write \mathbf{CHNL} . Define

$$\mathbf{NSUP} = \text{Sync}(\mathbf{SUP}'_1, \mathbf{SUP}_2); \quad (25)$$

then

$$\mathbf{SUP}' = \text{Sync}(\mathbf{NSUP}, \mathbf{CHNL}). \quad (26)$$

As before, write $\Sigma' = \Sigma \cup \{r'\}$, let $P : \Sigma'^* \rightarrow \Sigma^*$ be the natural projection of Σ'^* to Σ^* , and define the new natural projection

$$P_{\mathbf{CHNL}} : \Sigma'^* \rightarrow \{r, r'\}^*. \quad (27)$$

Proposition 2 *Given \mathbf{NSUP} in (25) and \mathbf{SUP}' in (26), suppose there exists a string $t = sr \in L(\mathbf{NSUP})$, such that*

- (i) $r \in \Sigma_u$,
- (ii) $(Ps)r \in L(\mathbf{G})$,
- (iii) $s \in L(\mathbf{SUP}')$.

Then $Pt \in L(\mathbf{SUP})$.

Remark 3 Condition (i) says that r is uncontrollable; condition (ii) means that the generated strings as viewed through P should belong to $L(\mathbf{G})$; condition (iii) states that s is generated by \mathbf{SUP}' .

Proof of Proposition 2. Because $s \in L(\mathbf{SUP}')$ and \mathbf{SUP} is delay-robust with respect to r , we have $Ps \in L(\mathbf{SUP})$. As \mathbf{SUP} is an optimal supervisor, $L(\mathbf{SUP})$ is controllable with respect to \mathbf{G} ; further,

$$(Ps)r \in L(\mathbf{SUP})\Sigma_u \cap L(\mathbf{G}) \subseteq L(\mathbf{SUP}).$$

Hence, $Pt = P(sr) = (Ps)r \in L(\mathbf{SUP})$. □

Suppose an implementation fault occurs, namely there exist $r \in \Sigma_u$ and $s \in L(\mathbf{SUP}')$ such that $sr \in L(\mathbf{NSUP})$, $\delta(0, P_{\mathbf{CHNL}}s) = 1$, but $sr \notin L(\mathbf{SUP}')$. By Proposition 2, $(Ps)r \in L(\mathbf{SUP})$; thus we see that the fault will not cause violation of the specification.

The above result can easily be extended to a more general case: Σ_{chn} contains more than one channeled event, i.e. any event, say $\alpha \in \Sigma_{chn}$, is transmitted from \mathbf{G}_2 to \mathbf{SUP}_1 , or from \mathbf{G}_1 to \mathbf{SUP}_2 , with indefinite delay via its channel \mathbf{CHNL}_α . For $i = 1, 2$, if \mathbf{SUP}_i imports event $\alpha \in \Sigma_{chn}$, let \mathbf{SUP}_i'' be the DES obtained from \mathbf{SUP}_i by replacing α by its signal event α' . Let

$$\mathbf{NSUP} = \text{Sync}(\mathbf{SUP}_1'', \mathbf{SUP}_2''), \quad (28)$$

and thus

$$\mathbf{SUP}' = \text{Sync}(\mathbf{NSUP}, \{\mathbf{CHNL}_\alpha | \alpha \in \Sigma_{chn}\}). \quad (29)$$

For \mathbf{NSUP} in (28), \mathbf{SUP}' in (29), and an uncontrollable event r (selected arbitrarily) in Σ_{chn} , suppose there exists a string $t = sr \in L(\mathbf{NSUP})$, such that $(Ps)r \in L(\mathbf{G})$ and $s \in L(\mathbf{SUP}')$. Then, as in Proposition 2, it can be shown that $Pt \in L(\mathbf{SUP})$ even if the transmission of r has not been completed by execution of r' , i.e. the implementation fault of executing r will not cause a violation of specification. Since r is arbitrarily selected, we conclude that the implementation fault of executing any uncontrollable event in Σ_{chn} is accepted by \mathbf{SUP}' .

Next we consider issue (2), namely whether the communication delay of an uncontrollable event could lead to violation of a modeling assumption. To formalize this issue we make the following definition.

Definition 4 Given \mathbf{NSUP} and \mathbf{SUP}' as in (25) and (26), let $r \in \Sigma_u$. If there exists $s \in L(\mathbf{SUP}')$ such that $sr \in L(\mathbf{NSUP})$, but $sr \notin L(\mathbf{SUP}')$, then we say that r is *blocked* by \mathbf{CHNL} .

Example 3 For illustration, let \mathbf{SUP}_1 and \mathbf{SUP}_2 be the generators shown in Fig. 9. Assume event 20 in \mathbf{SUP}_2 is exported to \mathbf{SUP}_1 , i.e., $r = 20$ and $r' = 120$; \mathbf{SUP}_1' is obtained by replacing 20 in \mathbf{SUP}_1 by 120. As shown in Fig. 10, $\mathbf{SUP}' = \text{Sync}(\mathbf{SUP}_1', \mathbf{CHNL}, \mathbf{SUP}_2)$ is easily verified to be delay-robust with respect to event 20. Define $\mathbf{NSUP} = \text{Sync}(\mathbf{SUP}_1', \mathbf{SUP}_2)$. Let $s = 20.21$; then $s.20 \in L(\mathbf{NSUP})$, but $s.20 \notin L(\mathbf{SUP}')$. Since $\mathbf{SUP}' = \text{Sync}(\mathbf{NSUP}, \mathbf{CHNL})$, event 20 is blocked by its channel \mathbf{CHNL} .

Event r will be blocked by \mathbf{CHNL} only when \mathbf{CHNL} is in state 1, at which r is not defined. So we create a new generator, which accepts sr when s leads \mathbf{CHNL} to state 1. On this basis the following algorithm is proposed to verify whether or not r is blocked by \mathbf{CHNL} in \mathbf{SUP}' .

Algorithm 1 (i) Marking all the states of \mathbf{NSUP} , we obtain a generator \mathbf{MNSUP} with

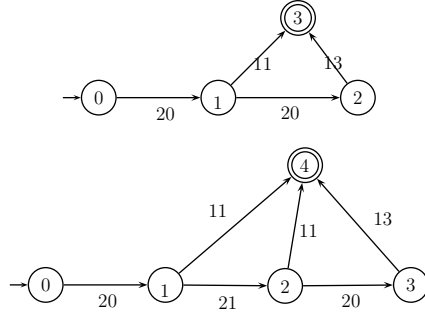
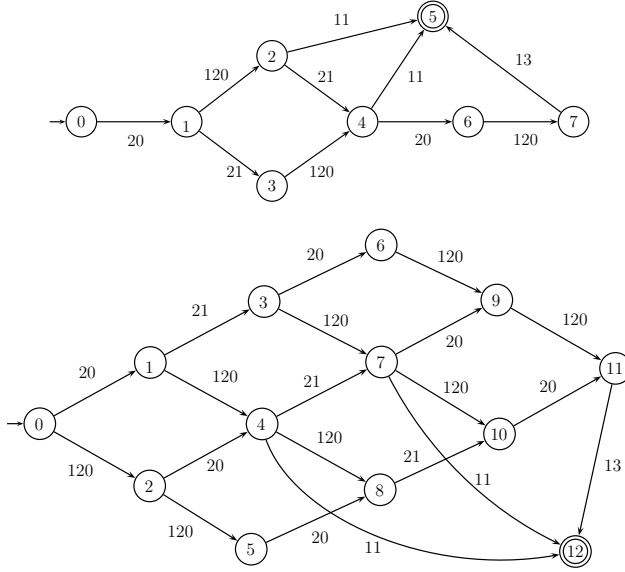
$$L_m(\mathbf{MNSUP}) = L(\mathbf{NSUP}). \quad (30)$$

(ii) Create a generator

$$\mathbf{NCHNL} = (\{0, 1, 2\}, \{r, r'\}, \delta_N, 0, \{2\})$$

where $\delta_N = [[0, r, 1], [1, r', 0], [1, r, 2], [2, r, 2]]$, as shown in Fig. 11.

(iii) Let $\mathbf{TEST} = \text{Sync}(\mathbf{MNSUP}, \mathbf{NCHNL})$, and $\mathbf{TTEST} = \text{Trim}(\mathbf{TEST})$.

Fig. 9 Example 3: SUP_1 and SUP_2 Fig. 10 Example 3: SUP' and NSUP

Note that **NCHNL** has the same alphabet $\{r, r'\}$ as **CHNL**. Also **MNSUP** does not change any event label in **NSUP**, thus its alphabet is Σ' . Hence,

$$\begin{aligned}
 L_m(\mathbf{TTEST}) &= L_m(\mathbf{TEST}) \quad (\text{by definition of } Trim) \\
 &= L_m(\mathbf{MNSUP}) || L_m(\mathbf{NCHNL}) \\
 &\quad (\text{by definition of } Sync) \\
 &= L_m(\mathbf{MNSUP}) \cap P_{\mathbf{CHNL}}^{-1} L_m(\mathbf{NCHNL}) \\
 &\quad (\{r, r'\} \subseteq \Sigma', P_{\mathbf{CHNL}} \text{ is defined by (27)}) \\
 &= L(\mathbf{NSUP}) \cap P_{\mathbf{CHNL}}^{-1} L_m(\mathbf{NCHNL}) \\
 &\quad (\text{by (30)}).
 \end{aligned}$$

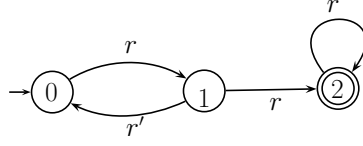


Fig. 11 NCHNL

Remark 4 Since $\mathbf{TTEST} = \text{Trim}(\mathbf{TEST})$, \mathbf{TTEST} is trim, i.e. \mathbf{TTEST} is reachable and coreachable [14]. If $L_m(\mathbf{TTEST})$ is empty, which implies that there are no marker states in \mathbf{TTEST} , then \mathbf{TTEST} is empty. On the other hand, if \mathbf{TTEST} is empty, obviously $L_m(\mathbf{TTEST})$ is empty. We conclude that \mathbf{TTEST} is empty iff $L_m(\mathbf{TTEST})$ is empty. So, \mathbf{TTEST} is nonempty iff

$$(\exists s \in (\Sigma \cup \{r'\})^*) s \in L(\mathbf{NSUP}) \cap P_{\mathbf{CHNL}}^{-1} L_m(\mathbf{NCHNL}).$$

We use the state set of \mathbf{TTEST} to identify whether or not r is blocked by \mathbf{CHNL} .

Theorem 3 *Let \mathbf{TTEST} be returned by Algorithm 1. Then r is blocked by \mathbf{CHNL} if and only if \mathbf{TTEST} is nonempty.*

Proof. (If) If event r is blocked by \mathbf{CHNL} , there exists $s \in L(\mathbf{SUP}')$ such that $sr \in L(\mathbf{NSUP})$, but $sr \notin L(\mathbf{SUP}')$. By $s \in L(\mathbf{SUP}')$, $s \in L(\mathbf{NSUP}) \cap P_{\mathbf{CHNL}}^{-1} L(\mathbf{CHNL})$. Thus, $P_{\mathbf{CHNL}}s \in L(\mathbf{CHNL})$. By $sr \notin L(\mathbf{SUP}')$ and $sr \in L(\mathbf{NSUP})$, $(P_{\mathbf{CHNL}}s)r \notin L(\mathbf{CHNL})$. According to the transition structure of \mathbf{CHNL} , $L(\mathbf{CHNL}) = (rr')^*$. We claim that $P_{\mathbf{CHNL}}s \in (rr')^*r$. Otherwise, $P_{\mathbf{CHNL}}s \in (rr')^*$. Since $P_{\mathbf{CHNL}}s \in L(\mathbf{CHNL})$ and $L(\mathbf{CHNL}) = (rr')^*$, $P_{\mathbf{CHNL}}(sr) \in (rr')^*r \subseteq (rr')^*rr' \subseteq (rr')^* = L(\mathbf{CHNL})$, which contradicts $(P_{\mathbf{CHNL}}s)r \notin L(\mathbf{CHNL})$. It follows that $(P_{\mathbf{CHNL}}s)r \in (rr')^*rr$, so $\delta_N(0, P_{\mathbf{CHNL}}(sr)) = 2$. Hence, $P_{\mathbf{CHNL}}(sr) \in L_m(\mathbf{NCHNL})$. It follows from $sr \in L(\mathbf{NSUP})$ that

$$sr \in L(\mathbf{NSUP}) \cap P_{\mathbf{CHNL}}^{-1} L_m(\mathbf{NCHNL}) = L_m(\mathbf{TTEST}).$$

Thus, \mathbf{TTEST} is nonempty, as claimed.

(Only if) If \mathbf{TTEST} is nonempty, there exists a string $t \in L_m(\mathbf{TTEST})$. Since $L_m(\mathbf{TTEST}) = L(\mathbf{NSUP}) \cap P_{\mathbf{CHNL}}^{-1} L_m(\mathbf{NCHNL})$, $t \in L(\mathbf{NSUP})$ and $P_{\mathbf{CHNL}}t \in L_m(\mathbf{NCHNL})$. By $P_{\mathbf{CHNL}}t \in L_m(\mathbf{NCHNL})$, and $L_m(\mathbf{NCHNL}) = (rr')^*r(rr^*)$, $P_{\mathbf{CHNL}}t \in (rr')^*r(rr^*)$, thus there exist $s \in (\Sigma \cup \{r'\})^*$ and $v \in \Sigma^*$ such that $t = srv$ with $P_{\mathbf{CHNL}}s \in (rr')^*r$ and $P_{\mathbf{CHNL}}v \in r^*$. We show that (i) $P_{\mathbf{CHNL}}s \in L(\mathbf{CHNL})$, and (ii) $P_{\mathbf{CHNL}}(sr) \notin L(\mathbf{CHNL})$. For (i), since $\delta(0, (rr')^*) = 0$ (δ is the transition function of \mathbf{CHNL}) and $\delta(0, (rr')^*r) = 1$, $\delta(0, P_{\mathbf{CHNL}}s) = \delta(0, (rr')^*r) = 1$, thus, $P_{\mathbf{CHNL}}s \in L(\mathbf{CHNL})$. For (ii),

$$\begin{aligned} \delta(0, P_{\mathbf{CHNL}}(sr)) &= \delta(0, (P_{\mathbf{CHNL}}s)r) \\ &= \delta(\delta(0, P_{\mathbf{CHNL}}s), r) \\ &= \delta(1, r). \end{aligned}$$

However, $\delta(1, r)$ is not defined in \mathbf{CHNL} . Hence, $P_{\mathbf{CHNL}}sr \notin L(\mathbf{CHNL})$.

Since $t \in L(\mathbf{NSUP})$, $s, sr \in L(\mathbf{NSUP})$. So,

$$s \in L(\mathbf{NSUP}) \cap P_{\mathbf{CHNL}}^{-1} L(\mathbf{CHNL}) = L(\mathbf{SUP}')$$

and

$$sr \notin L(\mathbf{NSUP}) \cap P_{\mathbf{CHNL}}^{-1}L(\mathbf{CHNL}) = L(\mathbf{SUP}').$$

By Definition 4, we conclude that r is blocked by \mathbf{CHNL} . \square

All the steps in Algorithm 1 can be implemented by standard software (e.g. [23]), and their validity and correctness are confirmed by Theorem 3.

Finally, we consider issue (2) in the general case described before, i.e. Σ_{chn} contains more than one communication event. For \mathbf{NSUP} in (28) and \mathbf{SUP}' in (29), let $r \in \Sigma_u \cap \Sigma_{chn}$. By (29), if there exists $s \in L(\mathbf{SUP}')$ such that $sr \in L(\mathbf{NSUP})$, but $sr \notin L(\mathbf{SUP}')$, then r is blocked by communication channels \mathbf{CHNL}_α ($\alpha \in \Sigma_{chn}$). However, for any $\alpha \in \Sigma_{chn} - \{r\}$, since r is not defined at any states of \mathbf{CHNL}_α , i.e. $s.r \in P_{\mathbf{CHNL}_\alpha}^{-1}L(\mathbf{CHNL}_\alpha)$ (the definition of $P_{\mathbf{CHNL}_\alpha}$ is similar to (27), replacing r and r' by α and α'), r will not be blocked by \mathbf{CHNL}_α . Hence, r is blocked by \mathbf{CHNL} (\mathbf{CHNL}_r is identical to \mathbf{CHNL}).

In Algorithm 1, replacing \mathbf{NSUP} and \mathbf{SUP}' by the DES in (28) and (29) respectively, and letting

$$\mathbf{TEST} = \text{Sync}(\mathbf{NSUP}, \mathbf{NCHNL}, \{\mathbf{CHNL}_\alpha | \alpha \in \Sigma_{chn} - \{r\}\}), \quad (31)$$

we obtain a new algorithm, say *Algorithm 2*. Let \mathbf{TTEST} be returned by Algorithm 2. As in Theorem 3, it can be proved that r is blocked by \mathbf{CHNL} if and only if \mathbf{TTEST} is nonempty, with the difference that $L(\mathbf{NSUP})$ is replaced by $L(\mathbf{NSUP}) \cap (\bigcap \{P_{\mathbf{CHNL}_\alpha}^{-1}L_m(\mathbf{CHNL}_\alpha) | \alpha \in \Sigma_{chn} - \{r\}\})$ in the appropriate instances.

Suppose r is uncontrollable. If \mathbf{TTEST} is empty, then

$$(\forall s \in L(\mathbf{SUP}')) sr \in L(\mathbf{NSUP}) \Rightarrow sr \in L(\mathbf{SUP}').$$

So r will not be blocked by \mathbf{CHNL} . In this case, \mathbf{SUP} is said to be ‘unbounded’ delay-robust with respect to r , similar to when r is controllable. If \mathbf{TTEST} is nonempty, there exists $s \in L(\mathbf{CHNL})$ such that $sr \in L(\mathbf{NSUP})$, but $sr \notin L(\mathbf{SUP}')$. By Proposition 2, although we have proved that the occurrence of r will not cause a fault which violates the specification, r is blocked by the channel. This could violate a modeling assumption since r is an uncontrollable event and should never be prohibited or delayed by an external agent. However, if the occurrence of r' is received by \mathbf{LOC}_1 before the next occurrence of r , the controllers will achieve global optimal nonblocking supervision. In this case, we say that \mathbf{SUP} is ‘bounded’ delay-robust with respect to r .

We illustrate these results by an example adapted from [14].

4 Example - WORKCELL

4.1 Model Description and Controller Design

WORKCELL consists of **ROBOT**, **LATHE** and **FEEDER**, with three buffers, **INBUF**, **LBUF** and **SBBUF**, connected as in Fig. 12. Labeled arrows denote synchronization on shared transitions (events) in the corresponding component DES.

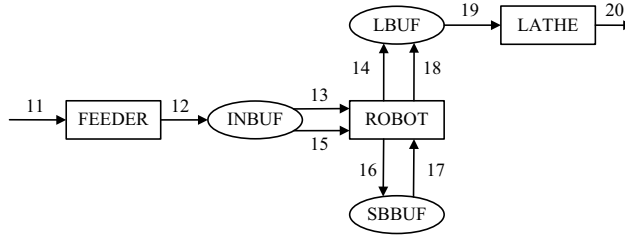


Fig. 12 WORKCELL

Table 1 Physical interpretation of events

Event label	Physical interpretation
11	FEEDER imports new part from infinite source
12	FEEDER loads new part in INBUF
13	ROBOT takes part from INBUF for loading into LBUF
14	ROBOT loads part from INBUF into LBUF
15	ROBOT takes part from INBUF for loading into SBBUF
16	ROBOT loads part from INBUF into SBBUF
17	ROBOT takes part from SBBUF for loading into LBUF
18	ROBOT loads part from SBBUF into LBUF
19	LATHE loads part from LBUF and starts working
20	LATHE exports finished part and returns to idle

WORKCELL operates as follows: **FEEDER** acquires a new part from an infinite source (event 11) then stores it (event 12) in a 2-slot buffer **INBUF**. **ROBOT** takes a new part from **INBUF** (event 13) and stores it (event 14) in a 1-slot buffer **LBUF**; if **LBUF** is already full, **ROBOT** may instead take a new part from **INBUF** (event 15) and store it (event 16) in a 1-slot ‘stand-by’ buffer **SBBUF**. If **LBUF** is empty and there’s already a part in **SBBUF**, **ROBOT** first unloads the part in **SBBUF** (event 17) and loads it in **LBUF** (event 18). If **LATHE** is idle and there exists a part in **LBUF**, **LATHE** takes that part and starts working on it (event 19), and when finished exports it and returns to idle (event 20). Event labels accord with [23]: odd-(resp. even-) numbered events are controllable (resp. uncontrollable). The physical interpretations of events are displayed in Table 1.

The specifications to be enforced are: 1) **SPEC₁** says that a buffer must not overflow or underflow; 2) **SPEC₂** says that **ROBOT** can load **SBBUF** (event sequence 15.16) only when **LBUF** is already full; 3) **SPEC₃** says that **ROBOT** can load **LBUF** directly from **INBUF** (event sequence 13.14) only when **SBBUF** is empty; otherwise it must load from **SBBUF** (event sequence 17.18).

The DES models of plant components and specifications are shown in Figs. 13 and 14.

We first compute the monolithic supervisor by a standard method (e.g. [14, 23]). The behavior of **WORKCELL** is the synchronous product of **FEEDER**, **ROBOT**, and **LATHE**. As **SPEC₁** is automatically incorporated in the buffer models, the total specification **SPEC** is the synchronous product of **INBUF**,

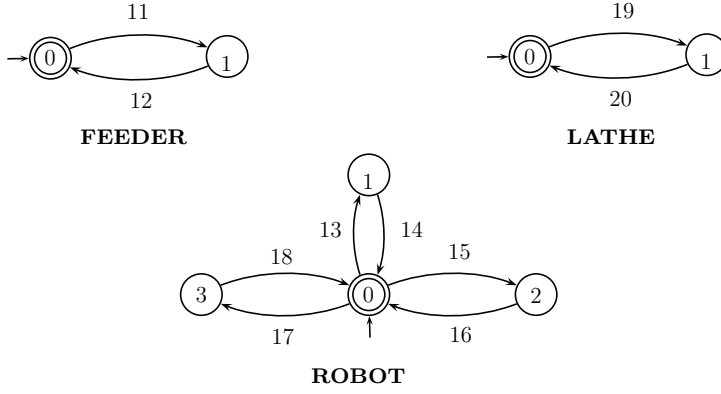


Fig. 13 Plant models to be controlled

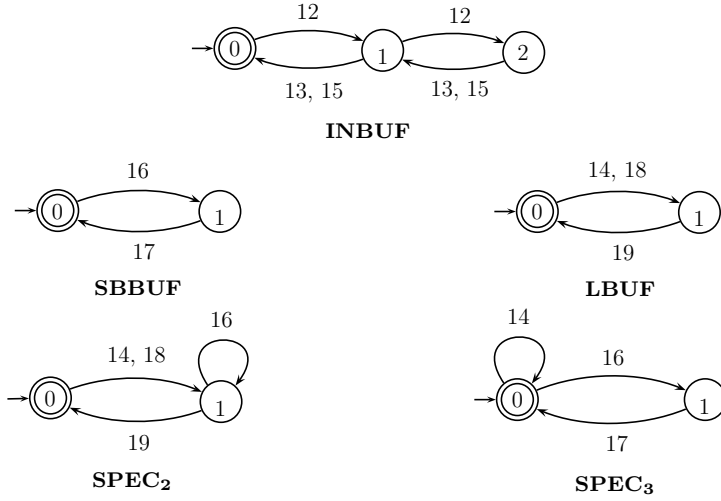


Fig. 14 Model of Specifications

LBUF, **SBBUF**, **SPEC₂**, and **SPEC₃**. The monolithic supervisor is

$$\mathbf{SUPER} = \text{Supcon}(\mathbf{WORKCELL}, \mathbf{SPEC})$$

with (state, transition) count (70, 153).

Next by use of procedure *Localize* [14, 23], we compute the localization of **SUPER** (in the sense of [5, 6]) to each of the three **WORKCELL** agents, to obtain local controllers **FEEDERLOC**, **ROBOTLOC** and **LATHELOC**, as shown in Fig. 15. The local controlled behaviors are

$$\begin{aligned} \mathbf{FEEDERSUP} &= \text{Sync}(\mathbf{FEEDER}, \mathbf{FEEDERLOC}), \\ \mathbf{ROBOTSUP} &= \text{Sync}(\mathbf{ROBOT}, \mathbf{ROBOTLOC}), \\ \mathbf{LATHESUP} &= \text{Sync}(\mathbf{LATHE}, \mathbf{LATHELOC}). \end{aligned}$$

From the transition structures shown in Fig. 15, we see that **FEEDERLOC** (**FEEDERSUP**) must import events 13 and 15 from **ROBOT**; **ROBOTLOC**

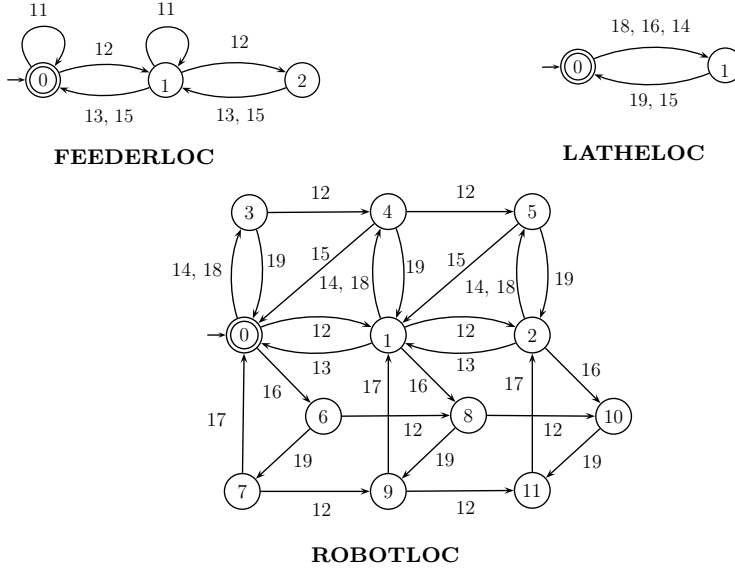


Fig. 15 Local Controller for each component



Fig. 16 CR13F and CR15F

(**ROBOTSUP**) must import events 12 from **FEEDER** and 19 from **LATHE**; and **LATHELOC** (**LATHESUP**) must import events 14, 15, 16, 18 from **ROBOT**.

4.2 Illustrative Cases

Based on the computed local controllers, we illustrate our new verification tools with the following cases.

Case 1 – Event 13

Taking **FEEDERLOC** for example, build a channel **CR13F**, as shown in Fig. 16, using a new event label 113 to represent the corresponding channel output; use this label to replace 13 in **FEEDERSUP** to obtain **FEEDERSUP'**, over the alphabet 11,12,113.

Now compute the channeled behavior **SUPER'** according to

$$\mathbf{SUPER}' = \text{Sync}(\mathbf{FEEDERSUP}', \mathbf{CR13F}, \mathbf{ROBOTSUP}, \mathbf{LATHESUP})$$

over the augmented alphabet $\{11, \dots, 20, 113\}$ and with (state, transition) count (112, 276). Next, to check delay-robustness we project **SUPER'** modulo supremal

quasi-congruence with nulled event 113, to get, say,

$$\begin{aligned} \mathbf{QCSUPER}' &:= \text{Supqc}(\mathbf{SUPER}', \text{Null}[113]) \\ &\quad (\text{deterministic, with size } (70, 153)) \end{aligned}$$

Finally we verify that $\mathbf{QCSUPER}'$ is isomorphic to \mathbf{SUPER} , and conclude that \mathbf{SUPER} is delay-robust with respect to the channeled communication of event 13 from \mathbf{ROBOT} to $\mathbf{FEEDERLOC}$. As a physical interpretation, consider the case where events 11, 12, 11, 12, 13 have occurred sequentially (i.e. there exist two parts in \mathbf{INBUF} and \mathbf{ROBOT} has taken a part from \mathbf{INBUF}) and $\mathbf{FEEDERSUP}'$ has not received the occurrence 113 of event 13. On the one hand, if $\mathbf{FEEDERSUP}'$ executes event 113 (i.e. it receives the occurrence of event 13), it will enable event 11 legally (according to \mathbf{SUPER}). On the other hand, if $\mathbf{FEEDERSUP}'$ does not execute event 113, then \mathbf{ROBOT} will load the part into \mathbf{LBUF} and take another part from \mathbf{INBUF} (execute event 15). So $\mathbf{FEEDERSUP}'$ can enable event 11 again, which is also legal according to \mathbf{SUPER} . Hence, in this case, the channeled system \mathbf{SUPER}' can run ‘correctly’ (no extra behavior violates the specification) and can ‘complete’ the given task (with the help of \mathbf{SBBUF}), i.e. the communication delay of event 13 is tolerable with respect to \mathbf{SUPER} . By the same method, one can verify that \mathbf{SUPER} is delay-robust with respect to events 12, 14, 15, 16, 18, 19 each taken separately.

Case 2 – Events 13 and 15

This case shows that \mathbf{SUPER} is delay-robust relative to the event set $\{13, 15\}$.

Similar to $\mathbf{CR13F}$, build another channel $\mathbf{CR15F}$, as shown in Fig. 16, using a new event label 115 to represent the corresponding channel output. Use labels 113, 115 to replace 13, 15 in $\mathbf{FEEDERSUP}$ to obtain $\mathbf{FEEDERSUP}'$, over the alphabet 11, 12, 113, 115.

We compute the channeled behavior \mathbf{SUPER}' according to

$$\begin{aligned} \mathbf{SUPER}' &= \text{Sync}(\mathbf{FEEDERSUP}', \mathbf{CR13F}, \mathbf{CR15F}, \\ &\quad \mathbf{ROBOTSUP}, \mathbf{LATHESUP}), \end{aligned}$$

over the augmented alphabet $\{11, \dots, 20, 113, 115\}$ and with (state, transition) count (148, 444). Next, to check delay-robustness we project \mathbf{SUPER}' modulo supremal quasi-congruence with nulled events 113, 115, to get, say,

$$\begin{aligned} \mathbf{QCSUPER}' &:= \text{Supqc}(\mathbf{SUPER}', \text{Null}[113, 115]) \\ &\quad (\text{deterministic, with size } (70, 153)) \end{aligned}$$

Finally $\mathbf{QCSUPER}'$ turns out to be isomorphic to \mathbf{SUPER} , and we conclude that \mathbf{SUPER} is delay-robust with respect to the channeled communication of events 13, 15 from \mathbf{ROBOT} to $\mathbf{FEEDERLOC}$. Briefly, the reason is that $\mathbf{FEEDERSUP}'$ will enable event 11 after it executes event 113 or 115, and \mathbf{ROBOT} will remain idle if no more parts are loaded into the system (i.e. event 11 cannot occur again).

Case 3 – Events 15 and 19

This case shows that \mathbf{SUPER} being delay-robust with respect to each event (15 or 19) taken separately does not imply that \mathbf{SUPER} is delay-robust with respect to the set $\{15, 19\}$.

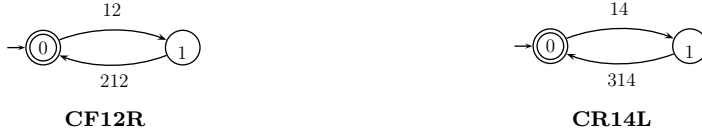


Fig. 17 CF12R and CR14L

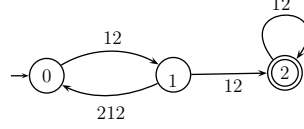


Fig. 18 NCF12R

Events 15 and 19 are shown, by Definition 2, to be delay critical when they are both delayed by an indefinite time. By tracking the working process, we show that communication delay of both events 15 and 19 may result in violation of **SPEC₂**. Consider the following case: events 11,12,11,12,13,14,19 have occurred sequentially, i.e. there exists one part in **INBUF**, **ROBOT** has loaded a part in **LBUF** and **LATHE** has taken the part from **LBUF** (i.e. **LBUF** is now empty). Since the transmission of event 19 is delayed unboundedly, if **ROBOT** does not know that **LATHE** has taken the part from **LBUF**, it may take a new part from **INBUF** (event 15) and load it into **SBBUF** (event 16) according to **ROBOTSUP'**. In this case, if there was no communication delay in transmitting event 15 (i.e. **LATHESUP** can observe the occurrence of event 15 instantly), event 15 will not occur when event 19 has occurred due to the synchronization of **LATHESUP** and **ROBOTSUP'** (according to **LATHESUP**, event 15 should not happen after event 19 occurred). Now event 15 is delayed indefinitely; thus **LATHESUP'** does not know the occurrence of event 15 before it receives the information (event 315) and event 15 (following with event 16) will occur before event 14 or 18 occurs (permitted by synchronization of **LATHESUP'** and **ROBOTSUP'**), i.e. the event sequence 11.12.11.12.13.14.19.15.16 occurs in the plant with communication delay, but violates **SPEC₂**. Hence, the event subset {15, 19} is delay-critical with respect to **SUPER**.

Case 4 – Event 12

This case shows that although the occurrence of (uncontrollable) event 12 may be blocked by its channel **CF12R**, as shown in Fig. 17, this will not violate the specifications.

Let $r = 12$ and $r' = 212$, and create **NCF12R** similarly to **NCHNL**, as shown in Fig. 18. Take

$$\mathbf{NSUPER} = \text{Sync}(\mathbf{FEEDERSUP}, \mathbf{ROBOTSUP}', \mathbf{LATHESUP})$$

and let **MNSUPER** be the generator obtained by marking all the states of **NSUPER**. Let

$$\mathbf{TTEST} = \text{Trim}(\text{Sync}(\mathbf{MNSUPER}, \mathbf{NCF12R}))$$

TTEST turns out to be nonempty, indeed, it contains $s = 11.12.11.12$. Physically, suppose 11, 12 and 11 have occurred sequentially, i.e., **FEEDER** has stored

a part in **INBUF** and taken another part (event 11). After that, **FEEDER** may store the part in **INBUF** (event 12, which is uncontrollable). Since s is in **L(SUPER)**, the occurrence of 12 will not violate global optimal supervision, as shown in Proposition 2. However, if **ROBOTSUP** does not receive the first occurrence of 12, then **CF12R** is at state 1, and thus cannot transmit the next occurrence of 12. So, in the channeled system **SUPER'**, event 12 is blocked by **CF12R**. If transmission of the first 12 is completed (i.e. event 212 occurs) before the second occurrence of event 12, then event 12 will not be blocked. In **SUPER**, only event 11 occurs between two occurrences of event 12; thus we say that **SUPER** is '1-bound'-delay-robust with respect to event 12.

Case 5 – Event 14

This case shows that the occurrence of event 14 will not be blocked by its channel **CR14L**, shown in Fig. 17.

Applying Algorithm 1 to event 14, the returned **TTEST** is empty. We conclude that event 14 will not be blocked by **CR14L**, and **SUPER** is unbounded-delay-robust with respect to 14. To illustrate the conclusion, we consider the following case: there exist two parts in **INBUF** (events 11,12,11,12 have occurred sequentially), and **ROBOT** has taken a part from **INBUF** (event 13) and placed it in **LBUF** (event 14). In Fig. 15, **FEEDERLOC** is at state 2 and is waiting for the occurrence of event 15 (**ROBOT** takes a part from **INBUF**); **ROBOTLOC** is at state 5 and is waiting for the occurrence of 19 (**LATHE** takes a part from **LBUF**) or waiting to enable event 15; and **LATHELOC** is at state 0 and is waiting for the occurrence of event 14. Only when the occurrence of event 14 has been received by **LATHELOC**, can event 15 and 19 be enabled by their corresponding controllers. Furthermore, event 14 cannot occur again until event 15 or event 19 occurs. Hence in this case the occurrence of event 14 is not blocked by its channel **CR14L**.

Case 6 – All communication events

When all communication events are subject to delay through channels (i.e. $\Sigma_{chn} = \Sigma_{com}$), it can be verified that delay-robustness of **SUPER** in the strong sense of Definition 3 (suitably extended to 3 agents) fails, i.e. **SUPER** fails to be delay-robust for distributed control by localization. Physically, when all the channeled events except 15 and 19 are received without delay, Case 6 is reduced to Case 3; thus **SUPER** is not delay-robust with respect to the set including all communication events, as confirmed by Corollary 1 in Sect. 3.

5 Conclusions and Future Work

In this paper we have studied distributed control obtained by supervisor localization on the relaxed assumption (compared to previous literature [5,6]) that inter-agent communication of selected 'communication events' (channeled events) may be subject to unknown time delays. For this distributed architecture we have identified a property of 'delay-robustness' which guarantees that the logical properties of our delay-free distributed control (i.e. the original DES specifications) continue to be enforced in the presence of delay, albeit with possibly degraded temporal behavior. We have shown that delay-robustness can be effectively tested, and that such tests serve to distinguish between events that are delay-critical and those

that are not. The case that an uncontrollable channeled event may be blocked by its communication channel is identified and for this a test procedure is provided. A simple workcell exemplifies the approach, showing how delay-robustness may depend on the subset of events subject to delay, and that a given event may be delay-critical for some choices of the delayed event subset but not for others.

With the definitions and tests reported here as basic tools, future work should include the investigation of global interconnection properties of a distributed system of DES which render delay-robustness more or less likely to be achieved. A quantitative approach involving timed discrete-event systems could also be an attractive extension.

Appendices

A Proof of Proposition 1

Recall that $\mathbf{SUP}' = (Y, \Sigma', \eta, y_0, Y_m)$. According to natural projection $P : \Sigma'^* \rightarrow \Sigma^*$ which maps $(\Sigma' - \Sigma)$ to ϵ , define $\eta' : Y \times \Sigma^* \rightarrow Pwr(Y)$ given by

$$\eta'(y, t) = \{\eta(y, s) \mid s \in \Sigma'^*, \eta(y, s)! \ \& \ Ps = t\}. \quad (32)$$

Let ρ be the supremal quasi-congruence on Y with respect to \mathbf{SUP}' , and define $P_\rho : Q \rightarrow Y/\rho = \bar{Y}$. As in ([14], Chapt. 6), $\mathbf{QCSUP}' = (\bar{Y}, \Sigma, \bar{\eta}, \bar{y}_0, \bar{Y}_m)$ is defined with $\bar{\eta} : \bar{Y} \times \Sigma^* \rightarrow Pwr(\bar{Y})$ given by

$$\bar{\eta}(\bar{y}, t) := \bigcup \{P_\rho(\eta'(y, t)) \mid P_\rho(y) = \bar{y}\}, \quad (33)$$

$$\bar{y}_0 = P_\rho(y_0) \text{ and } \bar{Y}_m = P_\rho(Y_m).$$

Proof. We must prove that \mathbf{QCSUP}' represents $PL_m(\mathbf{SUP}')$ and is a canonical generator.

(1) We show that \mathbf{QCSUP}' represents $PL_m(\mathbf{SUP}')$, i.e.,

$$L_m(\mathbf{QCSUP}') = PL_m(\mathbf{SUP}')$$

and

$$L(\mathbf{QCSUP}') = PL(\mathbf{SUP}').$$

(i) $L(\mathbf{QCSUP}') \subseteq PL(\mathbf{SUP}')$

Let $t \in L(\mathbf{QCSUP}')$. We prove by induction that $t \in PL(\mathbf{SUP}')$.

Base step: $t = \epsilon \in PL(\mathbf{SUP}')$ trivially.

Inductive step: Suppose $t \in L(\mathbf{QCSUP}')$, $t \in PL(\mathbf{SUP}')$, and $t\alpha \in L(\mathbf{QCSUP}')$; we must prove $t\alpha \in PL(\mathbf{SUP}')$.

Since $t\alpha \in L(\mathbf{QCSUP}')$, we have $\bar{\eta}(\bar{y}_0, t)!$ and $\bar{\eta}(\bar{y}_0, t\alpha)!$. So, $(\exists \bar{y} \in \bar{Y}) \bar{y} = \bar{\eta}(\bar{y}_0, t) \ \& \ \bar{\eta}(\bar{y}, \alpha)!$. We have $\bar{y}_0 = P_\rho(y_0)$. Since $t \in PL(\mathbf{SUP}')$, $(\exists s \in L(\mathbf{SUP}')) Ps = t$, i.e. $\eta(y_0, s)!$. So, $\eta(y_0, s) \in \eta'(y_0, t)$, i.e., $\eta'(y_0, t) \neq \emptyset$. Thus, $\bar{y} = P_\rho\eta'(y_0, t)$ because \mathbf{QCSUP}' is deterministic. Since $\bar{\eta}(\bar{y}, \alpha)!$ and $\eta(y_0, t) \neq \emptyset$, there exists $y \in \eta'(y_0, t)$ such that $\bar{\eta}(\bar{y}, \alpha) = P_\rho\eta'(y, \alpha)$. Hence, $\eta'(y_0, t\alpha)!$. However, according to (32)

$$\eta'(y_0, t\alpha) = \{\eta(y_0, s) \mid s \in \Sigma^*, \eta(y_0, s)!, Ps = t\alpha\}.$$

Thus, $(\exists s \in L(\mathbf{SUP}')) Ps = t\alpha$, so $t\alpha \in PL(\mathbf{SUP}')$.

(ii) $PL(\mathbf{SUP}') \subseteq L(\mathbf{QCSUP}')$

Let $t \in PL(\mathbf{SUP}')$; we show that $t \in L(\mathbf{QCSUP}')$.

Base step: $t = \epsilon \in L(\mathbf{QCSUP}')$ trivially.

Inductive step: Supposing $t \in PL(\mathbf{SUP}')$, $t \in L(\mathbf{QCSUP}')$, and $t\alpha \in PL(\mathbf{SUP}')$, we show $t\alpha \in L(\mathbf{QCSUP}')$.

Since $t \in PL(\mathbf{SUP}')$ and $t \in L(\mathbf{QCSUP}')$, $\eta'(y_0, t) \neq \emptyset$, $\bar{\eta}(\bar{y}_0, t)!$; letting $\bar{y} = \bar{\eta}(\bar{y}_0, t)$, then $\bar{y} = P_\rho \eta'(y_0, t)$ because \mathbf{QCSUP}' is deterministic. Since $t\alpha \in PL(\mathbf{SUP}')$, there exists $s' \in L(\mathbf{SUP}')$, i.e. $\eta(y_0, s')!$ such that $Ps' = t\alpha$; thus

$$\begin{aligned} & \bigcup \{\eta'(y', \alpha) | y' \in \eta'(y_0, t)\} \\ &= \bigcup \{\eta'(y', \alpha) | s \in \Sigma'^*, y' = \eta(y_0, s), Ps = t\} \quad (\text{according to (32)}) \\ &= \{\eta((\eta(y_0, s), v)) | v \in \Sigma'^*, \eta(\eta(y_0, s), v)!, Ps = t, Pv = \alpha\} \\ &= \{\eta(y_0, sv) | sv \in \Sigma'^*, \eta(y_0, sv)!, P(sv) = t\alpha\} \\ &\neq \emptyset \quad (\text{since } \eta(y_0, s')! \text{ and } Ps' = t\alpha), \end{aligned}$$

i.e. there exists $y \in \eta'(y_0, t)$ such that $\eta'(y, \alpha)!$. Then, $P_\rho y = \bar{y}$ due to $\bar{y} = P_\rho \eta'(y_0, t)$. Hence, $\bar{\eta}(\bar{y}, \alpha) = P_\rho \eta'(y, \alpha) \neq \emptyset$, i.e., $\bar{\eta}(\bar{y}, \alpha)!$. So, $t\alpha \in L(\mathbf{QCSUP}')$.

(iii) $L_m(\mathbf{QCSUP}') \subseteq PL_m(\mathbf{SUP}')$

For any $t \in \Sigma^*$, if $t \in L_m(\mathbf{QCSUP}')$, then $(\exists \bar{y} \in \bar{Y}) \bar{y} = \bar{\eta}(\bar{y}_0, t) \& \bar{y} \in \bar{Y}_m$. By (i), we conclude that $t \in PL(\mathbf{SUP}')$. Thus, $\eta'(y_0, t) \neq \emptyset$. Because \mathbf{QCSUP}' is deterministic, we know that $\bar{y} = P_\rho \eta'(y_0, t)$. So, $P_\rho \eta'(y_0, t) \in \bar{Y}_m$. Further, $\eta'(y_0, t) \cap Y_m \neq \emptyset$, i.e., there exists $s \in \Sigma'^*$ such that $\eta(y_0, s)!$ & $\eta(y_0, s) \in Y_m$ & $Ps = t$. Hence, $s \in L_m(\mathbf{SUP}')$, thus $t = Ps \in PL_m(\mathbf{SUP}')$.

(iv) $PL_m(\mathbf{SUP}') \subseteq L_m(\mathbf{QCSUP}')$

For any $t \in \Sigma^*$, if $t \in PL_m(\mathbf{SUP}')$, then $\eta'(y_0, t)!$ & $\eta'(y_0, t) \cap Y_m \neq \emptyset$. By (ii), $t \in L(\mathbf{QCSUP}')$, i.e., $(\exists \bar{y} \in \bar{Y}) \bar{\eta}(\bar{y}_0, t)!$ & $\bar{y} = \bar{\eta}(\bar{y}_0, t)$. Since \mathbf{QCSUP}' is deterministic, $\bar{y} = P_\rho \eta'(y_0, t)$. We conclude that $P_\rho \eta'(y_0, t) \in \bar{Y}_m$ from $\eta'(y_0, t) \cap Y_m \neq \emptyset$. Hence, $\bar{y} \in \bar{Y}_m$, i.e., $t \in L_m(\mathbf{QCSUP}')$.

2. We prove that \mathbf{QCSUP}' is a canonical(minimal-state) generator.

Let ν be a congruence on \bar{Y} defined according to: $\bar{y} \equiv \bar{y}' \pmod{\nu}$ provided

(i) $(\forall t \in \Sigma^*) \bar{\eta}(\bar{y}, t) \Leftrightarrow \bar{\eta}(\bar{y}', t)!$

(ii) $(\forall t \in \Sigma^*) \bar{\eta}(\bar{y}, t) \in \bar{Y}_m \Leftrightarrow \bar{\eta}(\bar{y}', t) \in \bar{Y}_m$.

With reference to ([14], Proposition 2.5.1), projection (mod ν) reduces \mathbf{QCSUP}' to a state-minimal generator.

Define $P_\nu : \bar{Y} \rightarrow \bar{Y}/\nu$ and write $\nu \circ \rho = \ker(P_\nu \circ P_\rho)$. Next we will prove that $\nu \circ \rho$ is a quasi-congruence on Y , i.e., for all $y, y' \in Y$,

$$P_\nu \circ P_\rho(y) = P_\nu \circ P_\rho(y') \Rightarrow (\forall \alpha \in \Sigma) P_\nu \circ P_\rho \eta(y, \alpha) = P_\nu \circ P_\rho \eta(y', \alpha).$$

Now

$$\begin{aligned} & P_\nu \circ P_\rho(y) = P_\nu \circ P_\rho(y') \\ &\Rightarrow P_\nu(P_\rho(y)) = P_\nu(P_\rho(y')) \\ &\Rightarrow P_\nu(\bar{\eta}(P_\rho(y)), \alpha) = P_\nu(\bar{\eta}(P_\rho(y')), \alpha) \\ &\quad (\text{cf. (ii) of Proposition 2.5.1 in [14]}) \\ &\Rightarrow P_\nu(\bar{\eta}(\bar{y}, \alpha)) = P_\nu(\bar{\eta}(\bar{y}', \alpha)) \\ &\Rightarrow P_\nu(P_\rho(\eta'(y, \alpha))) = P_\nu(P_\rho(\eta'(y', \alpha))) \\ &\Rightarrow P_\nu \circ P_\rho \eta'(y, \alpha) = P_\nu \circ P_\rho \eta'(y', \alpha) \end{aligned}$$

Hence, $\nu \circ \rho$ is a quasi-congruence on Y . Obviously, $\nu \circ \rho$ is coarser than ρ . However, ρ is the supremal quasi-congruence on Y , so for any $y, y' \in Y$, if $P_\nu(P_\rho(y)) = P_\nu(P_\rho(y'))$, i.e., $(y, y') \in \nu \circ \rho$, then $(y, y') \in \rho$, which means that $P_\rho(y) = P_\rho(y')$. Hence, $\nu = \perp$ (namely all its cells are singletons).

We have shown that \mathbf{QCSUP}' is a canonical generator. \square

B Proofs of Lemmas 1 - 3

To illustrate the proof idea of Lemmas 1-3, we consider a simpler case where $R_2 = \{\alpha, \beta\}$. As in Remark 2, $\mathbf{SUP}'_{A_{i+1}}$ is treated as the channeled behavior of \mathbf{SUP}'_{A_i} with event α_{i+1} being the channeled event. So, the general case of Lemmas 1-3 can be proved by the same method.

Table 2 Definition of each DES

DES	Alphabet	Definition
SUP₁	$\Sigma_{\text{SUP}_1} = \Sigma'_1$	$\text{Sync}(\mathbf{G}_1, \mathbf{LOC}_1)$
SUP₂	$\Sigma_{\text{SUP}_2} = \Sigma'_2$	$\text{Sync}(\mathbf{G}_2, \mathbf{LOC}_2)$
SUP	$\Sigma_{\text{SUP}} = \Sigma$	$\text{Sync}(\text{SUP}_1, \text{SUP}_2)$
SUP'₁	$\Sigma_{\text{SUP}'_1} = \Sigma'_1 \cup \{\alpha'\} - \{\alpha\}$	Replacing α in SUP₁ by α'
CH_α	$\Sigma_{\text{CH}_\alpha} = \{\alpha, \alpha'\}$	two state channel to transmit α
SUP'	$\Sigma_{\text{SUP}'} = \Sigma \cup \{\alpha'\}$	$\text{Sync}(\text{SUP}'_1, \text{CH}_\alpha, \text{SUP}_2)$
SUP''₁	$\Sigma_{\text{SUP}''_1} = \Sigma'_1 \cup \{\alpha', \beta'\} - \{\alpha, \beta\}$	Replacing α and β in SUP₁ by α' and β'
CH_β	$\Sigma_{\text{CH}_\beta} = \{\beta, \beta'\}$	two state channel to transmit β
NSUP'	$\Sigma_{\text{NSUP}'} = \Sigma \cup \{\alpha', \beta'\}$	$\text{Sync}(\text{SUP}''_1, \text{CH}_\alpha, \text{CH}_\beta, \text{SUP}_2)$

Table 3 Definition of natural projections

Natural Projection	Nullified events
$P_1 : \Sigma_{\text{SUP}}^* \rightarrow \Sigma_{\text{SUP}_1}^*$	$\Sigma - \Sigma'_1$
$P_2 : \Sigma_{\text{SUP}}^* \rightarrow \Sigma_{\text{SUP}_2}^*$	$\Sigma - \Sigma'_2$
$P'_1 : \Sigma_{\text{SUP}'}^* \rightarrow \Sigma_{\text{SUP}'_1}^*$	$\Sigma \cup \{\alpha\} - \Sigma'_1 (= \Sigma \cup \alpha' - (\Sigma'_1 \cup \{\alpha'\} - \{\alpha\}))$
$P'_2 : \Sigma_{\text{SUP}'}^* \rightarrow \Sigma_{\text{SUP}_2}^*$	$\Sigma \cup \{\alpha'\} - \Sigma'_2$
$P'_{\text{CH}_\alpha} : \Sigma_{\text{SUP}'}^* \rightarrow \Sigma_{\text{CH}_\alpha}^*$	$\Sigma - \{\alpha\} (= \Sigma \cup \{\alpha'\} - \{\alpha, \alpha'\})$
$P_{\alpha'} : \Sigma_{\text{SUP}'}^* \rightarrow \Sigma_{\text{SUP}}^*$	$\alpha' (= \Sigma \cup \{\alpha'\} - \Sigma)$
$P'_{N,1} : \Sigma_{\text{NSUP}'}^* \rightarrow \Sigma_{\text{SUP}'_1}^*$	$\Sigma \cup \{\alpha, \beta\} - \Sigma'_1$ $(= \Sigma \cup \{\alpha', \beta'\} - (\Sigma'_1 \cup \{\alpha', \beta'\} - \{\alpha, \beta\}))$
$P'_{N,2} : \Sigma_{\text{NSUP}'}^* \rightarrow \Sigma_{\text{SUP}_2}^*$	$\Sigma \cup \{\alpha', \beta'\} - \Sigma'_2$
$P'_{N, \text{CH}_\alpha} : \Sigma_{\text{NSUP}'}^* \rightarrow \Sigma_{\text{CH}_\alpha}^*$	$\Sigma \cup \{\beta'\} - \{\alpha\} (= \Sigma \cup \{\alpha', \beta'\} - \{\alpha, \alpha'\})$
$P'_{N, \text{CH}_\beta} : \Sigma_{\text{NSUP}'}^* \rightarrow \Sigma_{\text{CH}_\beta}^*$	$\Sigma \cup \{\alpha'\} - \{\beta\} (= \Sigma \cup \{\alpha', \beta'\} - \{\beta, \beta'\})$
$P_{\beta'} : \Sigma_{\text{NSUP}'}^* \rightarrow \Sigma_{\text{SUP}'}^*$	$\{\beta'\} (= \Sigma \cup \{\alpha', \beta'\} - (\Sigma \cup \{\alpha'\}))$
$P_{\alpha'\beta'} : \Sigma_{\text{NSUP}'}^* \rightarrow \Sigma_{\text{SUP}}^*$	$\{\alpha', \beta'\} (= \Sigma \cup \{\alpha', \beta'\} - \Sigma)$
$P'_{11} : \Sigma_{\text{SUP}'}^* \rightarrow (\Sigma_{\text{SUP}'} - \{\alpha\})^*$	α
$P'_{12} : (\Sigma_{\text{SUP}'} - \{\alpha\})^* \rightarrow \Sigma_{\text{SUP}'_1}^*$	$\Sigma - \{\Sigma'_1\} (= \Sigma \cup \{\alpha'\} - \{\alpha\} - (\Sigma'_1 \cup \{\alpha'\} - \{\alpha\}))$
$P_\beta : \Sigma_{\text{NSUP}'}^* \rightarrow (\Sigma_{\text{NSUP}'} - \{\beta\})^*$	β

Let α, β be the channeled events imported by **LOC₁** from **G₂** and transmitted by **CH_α** and **CH_β** respectively. Let α' (resp. β') be the output for α (resp. β) in **CH_α** (resp. **CH_β**), representing that **LOC₁** (**SUP₁** as well) receives the occurrence of α (resp. β) in **G₂**. Replacing all α in **SUP₁** by α' , we obtain **SUP'₁**; then

$$\text{SUP}' = \text{Sync}(\text{SUP}'_1, \text{CH}_\alpha, \text{SUP}_2). \quad (34)$$

Replacing all α and β in **SUP₁** by α' and β' correspondingly, we obtain **SUP''₁**; and write

$$\text{NSUP}' = \text{Sync}(\text{SUP}''_1, \text{CH}_\alpha, \text{CH}_\beta, \text{SUP}_2), \quad (35)$$

or equivalently

$$\text{NSUP}' = \text{Sync}(\text{SUP}''_1, \text{CH}_\beta, \text{Sync}(\text{SUP}_2, \text{CH}_\alpha)). \quad (36)$$

The definition of each DES and its alphabet is itemized in Table 2 and the definition of each natural projection is in Table 3.

To prove the lemmas, we first explain the relationship between **SUP₁** and **SUP'₁** by Lemmas 4 and 5: let $s = s_1\alpha...s_k\alpha s_{k+1}$ where $s_i \in (\Sigma - \{\alpha\})^*$ and $t = s_1\alpha'...s_k\alpha' s_{k+1}$; as shown in Fig. 19, $s \in P_1^{-1}L(\text{SUP}_1)$ is equivalent to $t \in P_1'^{-1}L(\text{SUP}'_1)$.

Lemma 4 1) Let $s \in P_1^{-1}L(\text{SUP}_1)$ and $k = \#\alpha(s)$ where $\#\alpha(s)$ denotes the number of occurrences of α in s . If $k = 0$, then $s \in P_1'^{-1}L(\text{SUP}'_1)$; if $k > 0$, write $s = s_1\alpha...s_k\alpha s_{k+1}$

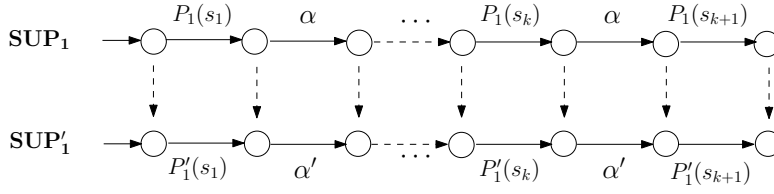


Fig. 19 $s \in P_1^{-1}L(\mathbf{SUP}_1) \Leftrightarrow t \in P_1'^{-1}L(\mathbf{SUP}'_1)$

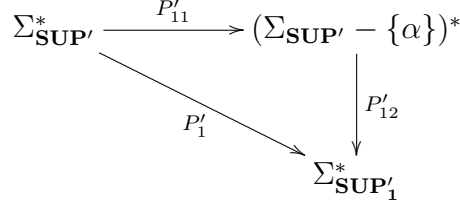


Fig. 20 $P'_1 = P'_{12}P'_{11}$

where $s_i \in (\Sigma - \{\alpha\})^*$; then $s_1\alpha' \dots s_k\alpha's_{k+1} \in P_1'^{-1}L(\mathbf{SUP}'_1)$ and $s_1\alpha\alpha' \dots s_k\alpha\alpha's_{k+1} \in P_1'^{-1}L(\mathbf{SUP}'_1)$.

2) Let $s \in P_1^{-1}L_m(\mathbf{SUP}_1)$. If $k = 0$, then $s \in P_1'^{-1}L_m(\mathbf{SUP}'_1)$; if $k > 0$, then $s_1\alpha' \dots s_k\alpha's_{k+1} \in P_1'^{-1}L_m(\mathbf{SUP}'_1)$ and $s_1\alpha\alpha' \dots s_k\alpha\alpha's_{k+1} \in P_1'^{-1}L_m(\mathbf{SUP}'_1)$.

Proof. For this proof, we define P'_{11} and P'_{12} as in Table 3; thus $P'_1 = P'_{12}P'_{11}$, as shown in Fig. 20. Statements 1) and 2) can be proved by the same method. We only prove 1), for the following two cases.

(i) $k = 0$. Since $k = 0$, $s \in (\Sigma - \{\alpha\})^*$. So $P_1(s) \in (\Sigma - \{\alpha\} - (\Sigma - \Sigma'_1))^* = (\Sigma_{\mathbf{SUP}_1} - \{\alpha\})^*$ and $P'_{12}(s) \in (\Sigma - \{\alpha\} - (\Sigma - \Sigma'_1))^* = (\Sigma_{\mathbf{SUP}_1} - \{\alpha\})^*$. It follows that $P'_{12}(s) = P_1(s)$. By $s \in P_1^{-1}L(\mathbf{SUP}_1)$, $P_1(s) \in L(\mathbf{SUP}_1)$. Because $k = 0$ and \mathbf{SUP}'_1 is obtained from \mathbf{SUP}_1 by replacing α with α' , $P_1(s) \in L(\mathbf{SUP}_1)$ implies that $P_1(s) \in L(\mathbf{SUP}'_1)$. Hence, $P'_{12}(s) = P_1(s) \in L(\mathbf{SUP}'_1)$. Due to $s \in (\Sigma - \{\alpha\})^* \subseteq (\Sigma \cup \{\alpha'\} - \{\alpha\})^*$, $P'_{11}(s) = s$. So $P'_1(s) = P'_{12}P'_{11}(s) = P'_{12}(s) \in L(\mathbf{SUP}'_1)$. So $s \in P_1'^{-1}L(\mathbf{SUP}'_1)$.

(ii) $k > 0$. Write $s = s_1\alpha' \dots s_k\alpha's_{k+1}$ where $s_i \in (\Sigma - \{\alpha\})^*$ ($i = 1, \dots, k+1$). So $P_1(s) = P_1(s_1)\alpha' \dots P_1(s_k)\alpha'P_1(s_{k+1})$. By $s \in P_1^{-1}L(\mathbf{SUP}_1)$, $P_1(s) \in L(\mathbf{SUP}_1)$; thus $P_1(s_1)\alpha' \dots P_1(s_k)\alpha'P_1(s_{k+1}) \in L(\mathbf{SUP}_1)$. As in (i), due to $\#\alpha(s_i) = 0$, $P'_{12}(s_i) = P_1(s_i)$ ($i = 1, \dots, k+1$). So, $P'_{12}(s_1)\alpha' \dots P'_{12}(s_k)\alpha'P'_{12}(s_{k+1}) \in L(\mathbf{SUP}_1)$; and thus, $P'_{12}(s_1)\alpha' \dots P'_{12}(s_k)\alpha'P'_{12}(s_{k+1}) \in L(\mathbf{SUP}'_1)$. Hence

$$\begin{aligned} P'_{12}(s_1\alpha' \dots s_k\alpha's_{k+1}) &= P'_{12}(s_1)P'_{12}\alpha' \dots P'_{12}(s_k)P'_{12}\alpha'P'_{12}(s_{k+1}) \\ &= P'_{12}(s_1)\alpha' \dots P'_{12}(s_k)\alpha'P'_{12}(s_{k+1}) \\ &\in L(\mathbf{SUP}'_1) \end{aligned}$$

Let $s' = s_1\alpha' \dots s_k\alpha's_{k+1}$. Since $s' \in (\Sigma_{\mathbf{SUP}'} - \{\alpha\})^*$, $P'_{11}(s') = s'$. So $P'_1(s') = P'_{12}P'_{11}(s') = P'_{12}(s') \in L(\mathbf{SUP}'_1)$; further $s' \in P_1'^{-1}L(\mathbf{SUP}'_1)$. Let $t = s_1\alpha\alpha' \dots s_k\alpha\alpha's_{k+1}$; then $P'_{11}(t) = s'$. Similarly, $t \in P_1'^{-1}L(\mathbf{SUP}'_1)$. \square

Lemma 5 1) Let $s \in P_1'^{-1}L(\mathbf{SUP}'_1)$, $\#\alpha(s) = 0$ and $k = \#\alpha'(s)$. If $k = 0$, then $s \in P_1^{-1}L(\mathbf{SUP}_1)$; if $k > 0$, write $s = s_1\alpha' \dots s_k\alpha's_{k+1}$ where $s_i \in (\Sigma - \{\alpha, \alpha'\})^*$; then $s_1\alpha \dots s_k\alpha s_{k+1} \in P_1^{-1}L(\mathbf{SUP}_1)$.

2) Let $s \in P_1'^{-1}L_m(\mathbf{SUP}'_1)$ and $\#\alpha(s) = 0$. If $k = 0$, then $s \in P_1^{-1}L_m(\mathbf{SUP}_1)$; if $k > 0$, then $s_1\alpha \dots s_k\alpha s_{k+1} \in P_1^{-1}L_m(\mathbf{SUP}_1)$.

Proof. Define P'_{11} and P'_{12} as in Table 3. Statements 1) and 2) can be proved by the same method. We only prove 1), for the following two cases.

(i) $k = 0$. As (i) in the proof of Lemma 4, $s \in P_1^{-1}L(\mathbf{SUP}_1)$ can be proved similarly.

(ii) $k > 0$. By $s \in P_1^{-1}L(\mathbf{SUP}_1)$, $P'_1(s) \in L(\mathbf{SUP}'_1)$. By $s \in (\Sigma_{\mathbf{SUP}'} - \{\alpha\})^*$, $P'_{11}(s) = s$; thus $P'_1(s) = P'_{12}P'_{11}(s) = P'_{12}(s)$. So $P'_{12}(s) \in L(\mathbf{SUP}'_1)$, i.e.

$$P'_{12}(s_1)\alpha' \dots P'_{12}(s_k)\alpha' P'_{12}(s_{k+1}) \in L(\mathbf{SUP}'_1).$$

As (i), $P'_{12}(s_i) = P_1(s_i)$ ($i = 1, \dots, k+1$). So, $P_1(s_1)\alpha' \dots P_1(s_k)\alpha' P_1(s_{k+1}) \in L(\mathbf{SUP}'_1)$. It follows that $P_1(s_1)\alpha \dots P_1(s_k)\alpha P_1(s_{k+1}) \in L(\mathbf{SUP}_1)$. Hence, $P_1(s_1\alpha \dots s_k\alpha s_{k+1}) \in L(\mathbf{SUP}_1)$, i.e. $s_1\alpha \dots s_k\alpha s_{k+1} \in P_1^{-1}L(\mathbf{SUP}_1)$. \square

Proof of Lemma 1. First we prove that $L(\mathbf{SUP}) \subseteq P_{\alpha'}L(\mathbf{SUP}')$; $L(\mathbf{SUP}') \subseteq P_{\beta'}L(\mathbf{NSUP}')$, $L_m(\mathbf{SUP}) \subseteq P_{\alpha'}L_m(\mathbf{SUP}')$ and $L_m(\mathbf{SUP}') \subseteq P_{\beta'}L_m(\mathbf{NSUP}')$ are proved similarly.

Let $s \in L(\mathbf{SUP})$ and $k = \#\alpha(s)$. Since $s \in L(\mathbf{SUP}) = P_1^{-1}L(\mathbf{SUP}_1) \cap P_2^{-1}L(\mathbf{SUP}_2)$, $s \in P_1^{-1}L(\mathbf{SUP}_1)$ and $s \in P_2^{-1}L(\mathbf{SUP}_2)$. First suppose $k = 0$. By Lemma 4, $s \in P_1^{-1}L(\mathbf{SUP}'_1)$. According to the definitions of P_2 and P'_2 , $s \in P_2^{-1}(\mathbf{SUP}_2)$ implies that $s \in P_2'^{-1}L(\mathbf{SUP}_2)$. Since $k = 0$, $P'_{\mathbf{CH}_\alpha} s = \epsilon$ (empty string), and thus $s \in P_{\mathbf{CH}_\alpha}'^{-1}L(\mathbf{CH}_\alpha)$. Hence,

$$s \in P_1'^{-1}L(\mathbf{SUP}'_1) \cap P_2'^{-1}L(\mathbf{SUP}_2) \cap P_{\mathbf{CH}_\alpha}'^{-1}L(\mathbf{CH}_\alpha) = L(\mathbf{SUP}').$$

Now suppose $k > 0$. Write $s = s_1\alpha \dots s_k\alpha s_{k+1}$ where $\#\alpha(s_i) = 0$ ($i = 1, \dots, k+1$). Let $t = s_1\alpha' \dots s_k\alpha' s_{k+1}$. By Lemma 4, $t \in P_1'^{-1}L(\mathbf{SUP}'_1)$. Because $\alpha' \notin \Sigma_{\mathbf{SUP}_2}$, $s \in P_2^{-1}L(\mathbf{SUP}_2)$ implies that $t \in P_2'^{-1}L(\mathbf{SUP}_2)$. Additionally, $P'_{\mathbf{CH}_\alpha} t = (\alpha\alpha')^k \in L(\mathbf{CH}_\alpha)$, i.e. $t \in P_{\mathbf{CH}_\alpha}'^{-1}L(\mathbf{CH}_\alpha)$, so $t \in L(\mathbf{SUP}')$ and thus $s = P_{\alpha'}t \in P_{\alpha'}L(\mathbf{SUP}')$. We conclude that $L(\mathbf{SUP}) \subseteq P_{\alpha'}L(\mathbf{SUP}')$, as required. \square

Proof of Lemma 2. In the simpler case where $R_2 = \{\alpha, \beta\}$, we must prove that if there exist $s \in L(\mathbf{SUP}')$ and $v \in \Sigma_{\mathbf{SUP}'}^*$ such that $\#\beta(v) = 0$ and $sv \notin L_m(\mathbf{SUP}')$, then there exists $t \in L(\mathbf{NSUP}')$ such that $P_{\beta'}t = s$ and for any $w \in \Sigma_{\mathbf{NSUP}'}^*$, if $P_{\beta'}w = v$, then $tw \notin L_m(\mathbf{NSUP}')$.

Let $\#\beta(s) = k$. We consider two cases: $k = 0$ and $k > 0$.

(i) $k = 0$. Let $t = s$; then $P_{\beta'}(t) = s$. By $s \in L(\mathbf{SUP}')$, $s \in P_1'^{-1}L(\mathbf{SUP}'_1)$, $s \in P_2'^{-1}L(\mathbf{SUP}_2)$, and $s \in P_{\mathbf{CH}_\alpha}'^{-1}L(\mathbf{CH}_\alpha)$. By Lemma 4, since $k = 0$, $t \in P_{N,1}'^{-1}L(\mathbf{SUP}_1'')$. By $s \in P_2'^{-1}L(\mathbf{SUP}_2)$, $P'_2(s) \in L(\mathbf{SUP}_2)$. Since $s \in (\Sigma_{\mathbf{SUP}'} - \{\beta\})^*$, $P'_{N,2}(s) = P'_2(s)$; thus $P'_{N,2}(s) \in L(\mathbf{SUP}_2)$. So $t = s \in P_{N,2}'^{-1}L(\mathbf{SUP}_2)$. Similarly, $t \in P_{N,\mathbf{CH}_\alpha}'^{-1}L(\mathbf{CH}_\alpha)$. Due to $\#\beta(t) = 0$, $P'_{N,\mathbf{CH}_\beta} t = \epsilon \in L(\mathbf{CH}_\beta)$. So $t \in P_{N,\mathbf{CH}_\beta}'^{-1}L(\mathbf{CH}_\beta)$. By (36), $t \in L(\mathbf{NSUP}')$. We claim that for any $w \in \Sigma_{\mathbf{NSUP}'}^*$, if $P_{\beta'}w = v$, then $tw \notin L_m(\mathbf{NSUP}')$. Otherwise, there exists $w \in \Sigma_{\mathbf{NSUP}'}^*$, $P_{\beta'}w = v$ and $tw \in L_m(\mathbf{NSUP}')$. Since $t = s \in (\Sigma_{\mathbf{SUP}'} - \{\beta\})^*$ ($\#\beta(s) = 0$) and $tw \in L_m(\mathbf{NSUP}')$, $\#\beta'(w) = 0$ and thus $\#\beta(w) = 0$. Furthermore, by $P_{\beta'}w = v$, $w = v$. Hence, $sv = tw \in L_m(\mathbf{NSUP}')$, i.e. $sv \in P_{N,1}'^{-1}L_m(\mathbf{SUP}_1'')$, $sv \in P_{N,2}'^{-1}L_m(\mathbf{SUP}_2)$ and $sv \in P_{N,\mathbf{CH}_\alpha}'^{-1}L(\mathbf{CH}_\alpha)$. By Lemma 5, $sv \in P_{N,1}'^{-1}L_m(\mathbf{SUP}_1'')$ implies that $sv \in P_1'^{-1}L_m(\mathbf{SUP}_1')$. Since $sv \in (\Sigma_{\mathbf{SUP}'} - \{\beta\})^*$, $P_{N,2}(sv) = P'_2(sv)$. So $sv \in P_{N,2}'^{-1}L_m(\mathbf{SUP}_2)$ implies that $sv \in P_{N,2}'^{-1}L_m(\mathbf{SUP}_2)$. Similarly, $sv \in P_{N,\mathbf{CH}_\alpha}'^{-1}L_m(\mathbf{CH}_\alpha)$. By (35), $sv \in L_m(\mathbf{SUP}')$; hence a contradiction.

(ii) $k > 0$. Write $s = s_1\beta \dots s_k\beta s_{k+1}$ where $s_i \in (\Sigma_{\mathbf{SUP}'} - \{\beta\})^*$ (thus $\#\beta(s_i) = 0$) ($i = 1, \dots, k+1$). Let $t = s_1\beta\beta' \dots s_k\beta\beta' s_{k+1}$; thus $P_{\beta'}t = s$. By $s \in L(\mathbf{SUP}')$, $s \in P_1'^{-1}L(\mathbf{SUP}'_1)$, $s \in P_2'^{-1}L(\mathbf{SUP}_2)$, and $s \in P_{\mathbf{CH}_\alpha}'^{-1}L(\mathbf{CH}_\alpha)$. By Lemma 4, $s \in P_1'^{-1}L(\mathbf{SUP}'_1)$ implies that $t \in P_{N,1}'^{-1}L(\mathbf{SUP}_1'')$. As in (i), $t \in P_{N,2}'^{-1}L(\mathbf{SUP}_2)$ and $t \in P_{N,\mathbf{CH}_\alpha}'^{-1}L(\mathbf{CH}_\alpha)$ can be proved. Additionally, $P'_{N,\mathbf{CH}_\beta}(t) = \beta\beta' \dots \beta\beta' \in (\overline{\beta\beta'})^* = L(\mathbf{CH}_\beta)$. So $t \in P_{N,\mathbf{CH}_\beta}'^{-1}L(\mathbf{CH}_\beta)$. By (36), $t \in L(\mathbf{NSUP}')$. We claim that for any $w \in \Sigma_{\mathbf{NSUP}'}^*$, if $P_{\beta'}w = v$, then $tw \notin L_m(\mathbf{NSUP}')$. Otherwise, there exists $w \in \Sigma_{\mathbf{NSUP}'}^*$, $P_{\beta'}w = v$ and $tw \in L_m(\mathbf{NSUP}')$. By $tw \in L_m(\mathbf{NSUP}')$, $P'_{N,\mathbf{CH}_\beta}(tw) \in L_m(\mathbf{CH}_\beta) = (\beta\beta')^*$. Since $P'_{N,\mathbf{CH}_\beta}(t) = \beta\beta' \dots \beta\beta'$, $P'_{N,\mathbf{CH}_\beta}w = (\beta\beta')^*$. By $\#\beta(P_{\beta'}(w)) = \#\beta(v) = 0$, $\#\beta(w) = \#\beta'(w) = 0$; thus $w = v$. So

$tw = tv \in L_m(\mathbf{NSUP}')$, and $tv = s_1\beta\beta' \dots s_k\beta\beta' s_{k+1}v$; thus $\#\beta(s_{k+1}v) = 0$. By (36) and $tv \in L_m(\mathbf{NSUP}')$, $tv \in P_{N,1}'^{-1}L_m(\mathbf{SUP}_1'')$, $tv \in P_{N,2}'^{-1}L(\mathbf{SUP}_2)$, and $tv \in P_{N,\mathbf{CH}_\alpha}'^{-1}L(\mathbf{CH}_\alpha)$. By $\Sigma_{\mathbf{SUP}_1''} = \{\Sigma_{\mathbf{SUP}_1} \cup \{\alpha', \beta'\} - \{\alpha, \beta\}\}$, $P_\beta(P_{N,1}'^{-1}L(\mathbf{SUP}_1'')) \subseteq P_{N,1}'^{-1}L(\mathbf{SUP}_1'')$ (where P_β is defined in Table 3); further, $tv \in P_{N,1}'^{-1}L_m(\mathbf{SUP}_1'')$ implies that $P_{\beta'}tv \in P_{N,1}'^{-1}L_m(\mathbf{SUP}_1'')$. Hence $P_\beta(tv) = s_1\beta' \dots s_k\beta' s_{k+1}v \in P_{N,1}'^{-1}L_m(\mathbf{SUP}_1'')$. Similarly, $sv = s_1\beta \dots s_k\beta s_{k+1}v \in P_2'^{-1}L_m(\mathbf{SUP}_2)$ and $sv \in P_{\mathbf{CH}_\alpha}'^{-1}L_m(\mathbf{CH}_\alpha)$. By Lemma 5, $sv \in P_1'^{-1}L_m(\mathbf{SUP}_1')$. By (35), $sv \in L_m(\mathbf{SUP}')$, a contradiction.

We conclude that there exists $t \in L(\mathbf{NSUP}')$ such that $P_{\beta'}t = s$ and for any $w \in \Sigma_{\mathbf{NSUP}'}^*$ if $P_{\beta'}w = \sigma$, then $tw \notin L(\mathbf{NSUP}')$. \square

Proof of Lemma 3. In the simpler case where $R_2 = \{\alpha, \beta\}$, we must prove that if there exist $s \in L(\mathbf{SUP}')$ and $\sigma \in \Sigma_{\mathbf{SUP}'}$ such that $s\sigma \notin L(\mathbf{SUP}')$, then there exists $t \in L(\mathbf{NSUP}')$ such that $P_{\beta'}t = s$ and for any $w \in \Sigma_{\mathbf{NSUP}'}^*$ if $P_{\beta'}w = \sigma$, then $tw \notin L(\mathbf{NSUP}')$.

Let $k = \#\beta(s)$. We consider two cases: $k = 0$ and $k > 0$.

(i) $k = 0$. Let $t = s$. As (i) in the proof of Lemma 2, $t \in L(\mathbf{NSUP}')$. We claim that for any $w \in \Sigma_{\mathbf{NSUP}'}^*$ if $P_{\beta'}w = \sigma$, then $tw \notin L(\mathbf{NSUP}')$. Otherwise, there exists $w \in \Sigma_{\mathbf{NSUP}'}^*$, $P_{\beta'}w = v$ and $tw \in L(\mathbf{NSUP}')$. If $\sigma \neq \beta$, then $tw \in (\Sigma_{\mathbf{SUP}'} - \{\beta\})^*$. Since $tw \in L(\mathbf{NSUP}')$, $P_{N,\mathbf{CH}_\beta}'tw \in L(\mathbf{CH}_\beta)$; thus $\#\beta'(tw) = 0$. Further, due to $P_{\beta'}w = \sigma$, $w = \sigma$. So $s\sigma = tw \in L(\mathbf{NSUP}')$. Similar to (i) in the proof of Lemma 2, $s\sigma \in L(\mathbf{SUP}')$. Hence, a contradiction. If $\sigma = \beta$, due to $P_{N,\mathbf{CH}_\beta}'tw \in L(\mathbf{CH}_\beta)$, then $w = \beta$ or $w = \beta\beta'$. If $w = \beta$, then $tw\beta' \in L(\mathbf{NSUP}')$. So, in both cases $t\beta\beta' \in L(\mathbf{NSUP}')$. Similar to (ii) in the proof of Lemma 2, $s\beta \in L(\mathbf{SUP}')$, which contradicts $s\sigma \notin L(\mathbf{SUP}')$.

(ii) $k > 0$. Write $s = s_1\beta \dots s_k\beta s_{k+1}$ where $s_i \in (\Sigma_{\mathbf{SUP}'} - \{\beta\})^*$ (thus $\#\beta(s_i) = 0$) ($i = 1, \dots, k+1$). Let $t = s_1\beta\beta' \dots s_k\beta\beta' s_{k+1}$. We claim that for any $w \in \Sigma_{\mathbf{NSUP}'}^*$ if $P_{\beta'}w = \sigma$, then $tw \notin L(\mathbf{NSUP}')$. Otherwise, there exists $w \in \Sigma_{\mathbf{NSUP}'}^*$, $P_{\beta'}w = v$ and $tw \in L(\mathbf{NSUP}')$. If $\sigma \neq \beta$, due to $P_{\beta'}(w) = \beta$, $\#\beta(w) = 0$. Since $P_{N,\mathbf{CH}_\beta}'t = \beta\beta' \dots \beta\beta' \in L_m(\mathbf{CH}_\beta)$, and $P_{N,\mathbf{CH}_\beta}'tw \in L(\mathbf{CH}_\beta)$, and $\#\beta'(w) = 0$. So $w = \sigma$, and thus $t\sigma = tw \in L(\mathbf{NSUP}')$. Similar to (ii) in the proof of Lemma 2, $s\sigma \in L(\mathbf{SUP}')$, a contradiction. If $\sigma = \beta$, as in (1), $t\beta\beta' \in L(\mathbf{NSUP}')$. Similar to (ii) in the proof of Lemma 2, $s\beta \in L(\mathbf{SUP}')$, again a contradiction.

We conclude that there exists $t \in L(\mathbf{NSUP}')$ such that $P_{\beta'}t = s$ and for any $w \in \Sigma_{\mathbf{NSUP}'}^*$ if $P_{\beta'}w = \sigma$, then $tw \notin L(\mathbf{NSUP}')$, as required. \square

C Delay-Robustness of Decentralized Controllers

Let \mathbf{G} be the DES to be controlled, and \mathbf{LOC}_1 and \mathbf{LOC}_2 be two decentralized controllers, which achieve global supervision with zero-delay communication. Let Σ_i , $\Sigma_{i\sigma}$ be the event set and observable event set of \mathbf{LOC}_i , respectively ($i = 1, 2$). Assume event $r \in \Sigma_1 \cap (\Sigma_{2\sigma} - \Sigma_{1\sigma})$, which is not observed by \mathbf{LOC}_1 , but is observed by \mathbf{LOC}_2 . Hence, r should be transmitted to \mathbf{LOC}_1 . We use the channel $\mathbf{C2r1}$, as shown in Fig. 1, to transmit r and use r' to represent that \mathbf{LOC}_1 receives the occurrence of r . Then, replacing r by r' , we obtain \mathbf{LOC}_1' . Let $\mathbf{SUP} = \text{Sync}(\mathbf{G}, \mathbf{LOC}_1, \mathbf{LOC}_2)$, $\mathbf{SUP}' = \text{Sync}(\mathbf{G}, \mathbf{LOC}_1', \mathbf{LOC}_2)$, and $\mathbf{QCSUP}' = \text{Supqc}(\mathbf{SUP}', \text{Null}[r'])$. Finally, by Theorem 1, if $\mathbf{SUP} \approx \mathbf{QCSUP}'$, \mathbf{SUP} is delay-robust with respect to r , or \mathbf{LOC}_1 and \mathbf{LOC}_2 achieve global supervision with unbounded delay communication.

References

1. R. Su and J.G. Thistle. A distributed supervisor synthesis approach based on weak bisimulation. In *Proc. 8th International Workshop on Discrete-Event Systems (WODES06)*, pages 64–69, Ann Arbor, MI, July 2006.
2. A. Mannani and P. Gohari. Decentralized supervisory control of discrete-event systems over communication networks. *IEEE Trans. on Automatic Control*, 53(2):547–559, March 2008.

3. P. Darondeau. Distributed implementation of Ramadge-Wonham supervisory control with Petri nets. In *Proc. 44th IEEE Conference on Decision and Control and 2005 European Control Conference. CDC-ECC'05*, pages 2107–2112, Seville, Spain, December 2005.
4. K. T. Seow, M. T. Pham, C. Ma, and M. Yokoo. Coordination planning: applying control synthesis methods for a class of distributed agents. *IEEE Trans. on Control Systems Technology*, 17(2):405–415, March 2009.
5. K. Cai and W. M. Wonham. Supervisor localization: a top-down approach to distributed control of discrete-event systems. *IEEE Trans. on Automatic Control*, 55(3):605–618, March 2010.
6. K. Cai and W.M. Wonham. Supervisor localization for large discrete-event systems: case study production cell. *International J. of Advanced Manufacturing Technology*, 50(9-12):1189–1202, October 2010.
7. M. Yeddes, H. Alla, and R. David. On the supervisory synthesis for distributed control of discrete event dynamic systems with communication delays. In *Proc. 1999 IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics*, pages 1–6, Cambridge, MA, September 1999.
8. G. Barrett and S. Lafortune. Decentralized supervisory control with communicating controllers. *IEEE Trans. on Automatic Control*, 45(9):1620–1638, September 2000.
9. S. Tripakis. Decentralized control of discrete-event systems with bounded or unbounded delay communication. *IEEE Trans. on Automatic Control*, 49(9):1489–1501, September 2004.
10. K.-H. Cho S.-J. Park. Decentralized supervisory control of discrete event systems with communication delays based on conjunctive and permissive decision structures. *Automatica*, 43(4):738–743, April 2007.
11. K. Hiraishi. On solvability of a decentralized supervisory control problem with communication. *IEEE Trans. on Automatic Control*, 54(3):468–480, March 2009.
12. P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event process. *SIAM J. Control and Optimization*, 25(1):206–230, January 1987.
13. P. J. Ramadge W. M. Wonham. On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization*, 25(3):637–659, May 1987.
14. W.M. Wonham. *Supervisory Control of Discrete-Event Systems*. Systems Control Group, ECE Dept, Univ. Toronto, Toronto, ON, Canada, July 2012. Available at <http://www.control.utoronto.ca/DES>.
15. L. Feng, K. Cai, and W. M. Wonham. A structural approach to the nonblocking supervisory control of discrete-event systems. *International J. of Advanced Manufacturing Technology*, 41(11):1152–1167, 2009.
16. K. Wong and W. M. Wonham. Modular control and coordination of discrete-event systems. *Discrete Event Dynamic Systems*, 8(3):247–297, October 1998.
17. R. Hill and D. Tilbury. Modular supervisory control of discrete-event systems with abstraction and incremental hierarchical construction. In *Proc. 8th International Workshop on Discrete-Event Systems (WODES06)*, pages 399–406, Ann Arbor, MI, July 2006.
18. R. Su, Jan H. van Schuppen, and Jacobus E. Rooda. Aggregative synthesis of distributed supervisors based on automaton abstraction. *IEEE Trans. on Automatic Control*, 55(7):1627–1640, July 2010.
19. J.Y. Udding. A formal model for defining and classifying delay-insensitive circuits and systems. *Distributed Computing*, 1:197–204, 1986.
20. H. Zhang. Delay Insensitive Networks. Master of math. thesis, Computer Science Dept., University of Waterloo, Waterloo, ON, Canada, 1997.
21. K.C. Wong and W.M. Wonham. On the computation of observers in discrete-event systems. *Discrete Event Dynamic Systems*, 14(1):55–107, January 2004.
22. L. Feng and W.M. Wonham. On the computation of natural observers in discrete-event systems. *Discrete Event Dynamic Systems*, 20(1):63–102, March 2010.
23. W.M. Wonham. *Design Software: XPTCT*. Systems Control Group, ECE Dept, Univ. Toronto, Toronto, ON, Canada, July 2012. Available at <http://www.control.utoronto.ca/DES>.
24. R. Milner. *Communication and Concurrency*. Prentice Hall, Englewood Cliffs, NJ, 1989.